VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Rozšíření řídícího software motorizované panoramatické hlavy

Extension of the Control Software of Motorized Panoramic Heads

2017 Bc. Tomáš Obadal

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Tomáš Obadal**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Rozšíření řídícího software motorizované panoramatické hlavy
Extension of the Control Software of Motorized Panoramic Heads

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem diplomové práce je rozšíření existujícího řídícího software motorizované panoramatické hlavy o aktivní rozhraní mezi panoramatickou hlavou a digitálním fotoaparátem. Dalším cílem je návrh a implementace aplikace pro tablety a mobilní telefony se systémem OS Android za účelem vzdáleného ovládání panoramatické hlavy a fotoaparátu.

1. Seznamte se s Picture Transfer Protocol (PTP) a možnostmi jeho využití pro komunikaci s digitálním fotoaparátem.
2. Seznamte se s problematikou návrhu a vývoje aplikací pro OS Android.
3. Popište komerční systémy s obdobnou funkcionalitou.
4. Implementujte rozhraní mezi panoramatickou hlavou a digitálním fotoaparátem.
5. Implementujte mobilní aplikaci k vzdálenému ovládání hlavy a fotoaparátu.
6. Proveďte důkladné testování za účelem zajištění maximální spolehlivosti software.
7. Vypracujte programátorskou dokumentaci.

Seznam doporučené odborné literatury:

[1]Photographic and imaging Manufactures Association, INC. : Photography – Electronic still picture imaging. Picture Transfer Protocol (PTP) for digital still photography devices, 2005, ISBN: 0-580 46685-X
[2] Hellman, E.: Android Programming: Pushing the Limits, Wiley, 2013, ISBN: 978-1118717370
[3] Iversen, J., Eierman, M.:Learning Mobile App Development: A Hands-on Guide to Building Apps with iOS and Android, Addison-Wesley Professional, 2013, ISBN: 978-0321947864

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. David Seidl, Ph.D.**

Datum zadání: 01.09.2015

Datum odevzdání: 28.04.2017

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 28. dubna 2017 .....

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.

V Ostravě 28. dubna 2017 .....

Rád bych poděkoval vedoucímu diplomové práce Ing.Davidu Seidlovi, Ph.D. za odbornou pomoc a konzultaci při vytváření této práce. Dále také rodině za bezmeznou podporu a pomoc při korekci výsledného textu.

Abstrakt

Práce se zabývá implementací rozšíření stávajícího řídícího software pro motorizovanou panoramatickou hlavu, kdy hlavní funkcí systému je pořizování panoramatických snímků ve velmi vysokém rozlišení. Rozšíření se skládá ze dvou hlavních částí. První se zabývá implementací aktivního rozhraní mezi řídící jednotkou a fotoaparátem, které zajistí zpětnou vazbu fotoaparátu, a tím přesnější a spolehlivější chod systému jako celku. Druhým významným rozšířením je pak možnost vzdáleného ovládání obou hlavních komponent - panoramatické hlavy a fotoaparátu. Díky tomu je možné použití i na místech, kde by to dříve bylo jen velmi obtíženě realizovatelné.

Klíčová slova: panoramatická fotografie, PTP, Raspberry PI, WebSocket

Abstract

The work deals with the implementation of the extension of the existing control software for motorized panoramic head. The main function is to capture panoramic images at very high resolution. Extension consists of two main parts. The first deals with the implementation of an active interface between the controller and camera to ensure camera feedback, and more accurate and reliable operation of the system as a whole. The second important extension is the possibility of remote control of two main components - the panoramic head and camera. This makes it possible to use them in places where it previously was very difficult.

Key Words: panoramatic photo, PTP, Raspberry PI, WebSocket

Obsah

Seznam použitých zkratek a symbolů

PTP	–	Picture Transfer Protocol
PTP/IP	–	Picture Transfer Protocol over IP
USB	–	Universal Serial Bus
EV	–	Exposure Value
HTML	–	HyperText Markup Language
CSS	–	Cascading Style Sheets
HTTP	–	Hypertext Transfer Protocol
HTTPS	–	Hypertext Transfer Protocol Secure
GUID	–	Globally Unique Identifier
SHA	–	Secure Hash Algorithm

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

Tato práce se nejdříve zaobírá problematikou panoramatické fotografie. Dále popisuje potřebný software a hardware k úspěšnému vytvoření panoramatického snímku a také představuje i hardware použitý pro běh implementovaného software. Hlavní částí diplomové práce je popis rozšiřujících částí řídícího software motorizované panoramatické hlavy, a to aktivního rozhraní mezi fotoaparátem a řídící jednotou, načež i vzdáleného ovládání. Potřeba těchto rozšíření vznikla na základě aktivního používání řídícího software a mají zajistit zvýšení užitné hodnoty software a zařízení, vyšší míru použitelnosti a dosažení lepších výstupních panoramat.

1.1 Panoramatická fotografie

Panoramatická fotografie je taková fotografie, která zaznamenává výrazně větší úhel záběru ve srovnání s běžnou fotografií, která je produkována dostupnými digitálními fotoaparáty dnešní doby. Za první panoramatickou fotografii je považován snímek rakouského občana Josepha Puchbergerva, kterému se v první polovině 19. století podařilo vytvořit panorama s úhlem záběru 150 °.
Panoramatická fotografie se dělí podle způsobu vytvoření:

- **Specializovaným panoramatických fotoaparátem nebo objektivem** – snímek je vytvořen pořízením jediného snímku. Používá se k tomu zařízení, které je primárně určeno k panoramatické fotografii nebo fotoaparáty s výměnnými objektivy, kde s použitím vhodného objektivu lze dosáhnout velkých úhlů záběru. Výhodou je rychlost pořízení a vyrovnaná expozice napříč snímkem. Nevýhodou tohoto způsobu vytvoření je pak nízké rozlišení a fotografie mnohdy vykazuje optické deformace způsobené stavbou objektivů.

- **Skládáním snímku s běžným úhlem záběru** – tento způsob pořízení panoramatické fotografie je dnes nejpoužívanější a pokud se v této práci mluví o panoramatické fotografii, je výhradně myšlen tento způsob vzniku. Téměř každý digitální fotoaparát nebo mobilní telefon dnešní doby obsahuje režim, který umožňuje vyfotografovat panoramatický snímek. Výsledný snímek vzniká tak, že je pořízeno několik fotografií s běžným úhlem záběru, které jsou poté za použití vhodný algoritmů složeny do jediného snímku s velkým úhlem záběru. Výhodou je možnost vzniku panoramata s velkým rozlišením, které obsahuje obrovské množství detailů. Nevýhodou jsou poté nároky na správné nastavení fotoaparátu a případné panoramatické hlavy tak, aby vznikl celistvý dojem jediné fotografie se správnou kompozicí. Tyto nároky se v přímé úměře zvyšují s požadovaným výsledným rozlišením panoramata.

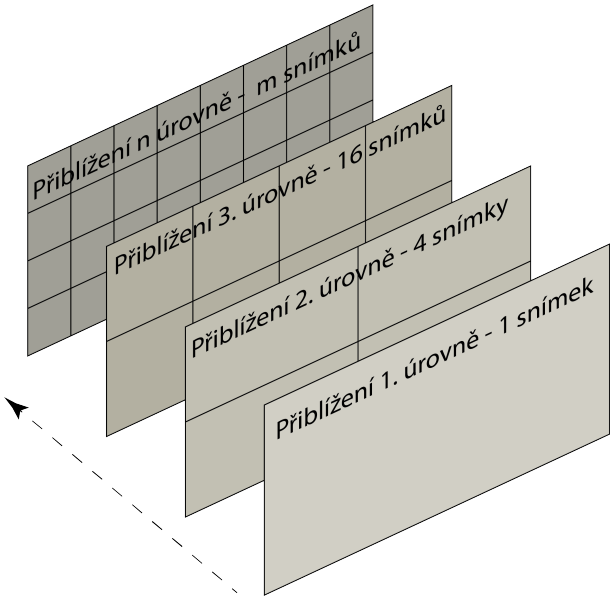
Častým problémem, který po pořízení panoramatické fotografie vzniká, je jeho správná projekce na monitorech počítačů, či na vytištěné fotografii. Svět, ve kterém je panorama snímáno, se pro pozorovatele jeví jako vnitřek koule. Monitor počítače nebo tištěná fotografie jsou ovšem

ploché. Podobný problém se musí například řešit při projekcí map světa, svět je kulatý, ale jeho projekce dvourozměrná. Proto aby zobrazování panoramat bylo pro člověka přirozené a nedocházelo ke zkreslení, musí být zvolena vhodná projekce. Existuje jich celá řada, ale typicky se používají tři základní:

- **Cylindrické panorama** - panoramatická fotografie je vykreslována na povrch válce, kdy pozorovatel stojí uvnitř. Působí přirozeně, pokud zorný úhel zabírá celý kruh, tedy 360 °.
- **Krychlové panorama** - panorama je vykresleno na šesti stěnách krychle. Používá se pro zobrazení při vertikálním úhlu větším, než 140 °. Pro diváka působí přirozeně zvláště v případech, kdy je na snímku zachycena pouze krajina.
- **Kulové panorama** - také nazývané sférické. Dominantní projekce, která je používána nejčastěji. Pokud panorama snímá maximálně možných 360/180 °, je tato projekce nejpřirozenější. Divák stojí uprostřed kupole, na kterou je promítán snímek.

1.2 Prezentování na internetu

Malé panoramatické snímky je možné prezentovat jako běžné snímky a sdílet je na běžných službách k tomu určených. Problém ovšem nastává, pokud je potřeba prezentovat panoramata ve velmi vysokém, často gigapixelovém rozlišení. Takový snímek může na disku počítače zabírat řádově desítky gigabajtů, a není proto možné je běžně dostupnými prostředky prezentovat. Kvůli tomu byl vyvinut způsob, který to umožňuje a přitom neklade velké nároky na internetové připojení a výkon počítače uživatele. Základní princip spočívá v tom, že si prohlížeč uživatele stáhne jen tu konkrétní část panoramata, na kterou se zrovna dívá. Autor panoramatické fotografie, předtím než jej nahraje na internet, ho rozdělí na vrstvy, kdy každá vrstva reprezentuje jednu úroveň přiblížení. Každá vrstva se pak dále rozdělí na menší části, kde je každý jednotlivý snímek výřezem dané vrstvy. Vrstvení ukazuje obrázek 1 (převzat z [5]). U gigapixelové fotografie pak ve výsledku mohou vzniknout statisíce snímků o průměrné velikosti 30KB. Ke správnému zobrazení je poté potřeba specializovaný přehrávač, který pro uživatele jednotlivé malé dílky sestaví do celistvého snímku. Výhodou použití specializovaného přehrávače pak může být možnost přidávat dodatečné informace o vyfotografovaných objektech, videa, apod.



Obrázek 1: Vrstvení snímků podle míry přiblížení

1.3 Hardware

K vytvoření panoramatického snímku je v základu potřeba pouze fotoaparát. Složením dvou fotografií s běžným úhlem záběru je možné dosáhnout vytvoření panoramatického snímku. Pro vyfocení většího panoramatu je velkým pomocníkem kvalitní fotografický stativ, který usnadní horizontální i vertikální posuv mezi jednotlivými snímky. Vytvořit ale tímto způsobem panorama s gigapixelovým rozlišením, je prakticky nemožné. K němu je zapotřebí vyfotit stovky až tisíce jednotlivých snímků, které musí být ve správném rozestupu a musí být vyfoceny dostatečně rychle tak, aby celková expozice složeného snímku byla jednotná. Je nutné použít fotoaparát s objektivem, který má velké ohnisko, typicky 200mm a více. Zorný úhel jednotlivých snímků je poté tak malý, že jednotlivé posuvy mezi snímky není možné zvládnout ručně. Je proto nutné použít specializovaný hardware, tzv. motorizovanou panoramatickou hlavu, která samotné focení stovek až tisíců snímků a jednotlivé posuvy, zvládne sám.

Motorizovanou panoramatickou hlavu jsem si vyvíjel a sestavil sám. Hlavní část stavby jsem provedl pro potřeby mé bakalářské práce [5] a pro stavbu vlastního řešení jsem se rozhodl z těchto důvodů:

- **neexistující programovatelné řešení** - neexistence opensource řešení, které by na existující hardware umožnilo vyvíjet vlastní software.
- **cena** - oproti komerčním produktům jsem byl schopen vytvořit zařízení za třetinové náklady.
- **funkce** - funkcionalita byla v konečném důsledku vyšší, než u komerčně nejdostupnějšího řešení, což se ještě zvýší po dokončení rozšíření.

Vlastní zařízení se skládá ze dvou hlavních komponentů:

- **Přístrojová skříň** - obsahuje hlavní řídící jednotku, plošný spoj, akumulátor, jsou do ní zapuštěna tlačítka a displej. Také je v ní ukryt jeden ze dvou krokových motorů.
- **Pohyblivé rameno** - to je tvořeno dvěma spojenými profily do tvaru písmene L. Na vertikálním profilu je připevněn nastavitelný držák fotoaparátu. Dva krokové motory pak umožňují pohyb v osách x a y, čímž je umožněno vyfotografovat panoramata v plném rozsahu až 360/180 °, čemuž muselo být přizpůsobeno vedení kabeláže, aby nedocházelo ke kroucení a pohyb nebyl v žádném směru omezen.

1.3.1 Řídící jednotka

Oproti předchozí verzi, kdy byla jako řídící jednotka použit mikrokontroler AT-mega od společnosti Atmel, jsem se rozhodl, že pro potřeby rozšíření použiji jako řídící jednotku Raspberry Pi 3, model B+. Důvody, které vedly k této výrazné změně byly následující:

- **nedostatečný výkon AT-mega** - pro potřeby rozšíření byl již výkon tohoto mikrokontroleru nedostatečný. Nový rozšiřující software vyžaduje běh více vláken a webového serveru, přenos fotografií, což by se stávajícím mikrokontrolerem bylo těžko dosažitelné.
- **cena** - přestože je Raspberry Pi 3 dražší, než předchozí řešení, cena je stále přijatelná a nabízí za ni výborné parametry pro potřeby mého rozšíření.
- **architektura** - Raspberry Pi 3 je postaven na 64 bitovém ARM procesoru, na kterém je možné provozovat celou řadu distribucí s bohatou uživatelskou základnou a nabídkou dostupných aplikací.
- **konektivita** - v základu je nabízena bezdrátová technologie Wi-Fi, přes kterou bude provozováno vzdálené ovládání, dále pak dostatečný počet I/O pinů pro připojení tlačítek, displeje a dalších periférií.
- **programovatelnost** - protože se jedná o univerzální zařízení, lze na něm programovat a spouštět programy v nejrůznějších programovacích jazycích, díky čemuž jsem mohl pokračovat v již hotové práci, která používala mikrokontroler Atmel.

1.4 Software

Po vytvoření jednotlivých snímků panoramata, u gigapixelové fotografie pak typicky stovky až tisíce, je potřeba využít specializovaného softwaru, který dokáže snímky analyzovat a spojit do jednotného panoramatu. Je klíčové, aby snímky byly fotografovány tak, že mezi sousedními je vždy překrytí. Typicky se používají hodnoty od 25% do 30% dle okolních podmínek a focené scény. Software pak dokáže v těchto překrytích nalézt společné prvky, typicky na kontrastních hranách, a dle toho je pak umístit na odpovídající místo v celkovém panoramatu.

Existuje celá řada komerčních produktů, kdy ty nejpokročilejší dokáží korigovat barevné odchylky mezi snímky, geometrické deformace nebo i běžné výrobní vady objektivů. U sestavování gigapixelových panoramat platí, že se jedná o výpočetně velice náročnou operaci, pro kterou je potřeba velmi výkonný hardware s dostatkem operační paměti a procesorového výkonu.

2 Rozhraní mezi panoramatickou hlavou a fotoaparátem

Hlavním cílem implementace rozhraní mezi panoramatickou hlavou a fotoaparátem byla možnost získat zpětnou vazbu od fotoaparátu, která výrazně zvýší spolehlivost systému jako celku a umožní fotografování scén, které s dřívějším řídícím software nebyla možná.

2.1 Picture Transfer Protocol

U všech nejvýznamějších značek výrobců fotoaparátů je dnes ke komunikaci mezi fotoaparátem a počítačem využit PTP protokol. Ten je dále u fotoaparátů obsahující Wi-Fi rozšířen o podporu PTP/IP, kdy je možné provozovat PTP protokol v běžném TCP/IP spojení. Každý výrobce si nad rámec standardu (verze 1.1 ISO 15740:2013) přidává své vlastní příkazy, vlastnosti a události. Protokol byl vyvinut Mezinárodní asociací obrazového průmyslu jako nástroj k přenášení obrázků z digitálních fotoaparátů do počítače, přímý tisk z fotoaparátu, a to bez nutnosti instalace ovladačů. Typickým prostředkem pro přenos je použito USB. Přenos je obousměrný.

2.1.1 Komunikační protokol

PTP standard umožňuje komunikaci mezi dvěma zařízeními. Ty jsou označovány jako Iniciátor (Initiator) a Respondent (Responder). Iniciátor je zařízení, které otevírá relaci a inicializuje požadované operace skrze vhodné transportní médium, typicky USB, dle jeho specifikace. Respondent je poté zařízení, které odpovídá na požadované operace tím, že odesílá požadovaná data, odpovědi nebo události. Zařízení mohou mít schopnosti být Iniciátorem, Respondentem nebo obojím. Většina dnešních digitálních fotoaparátů umožňuje obě tyto možnosti, kdy při spojení s počítačem se kamera chová jako Respondent, ale umí být i Iniciátor, a to při přímém spojení s tiskárnou.

2.1.2 Datové typy

Tabulka 1 shrnuje standardní datové typy a název pak naznačuje typický význam.

2.1.2.1 Datacodes

Datacodes jsou reprezentovány 16 bitovým unsigned integrem (UINT16). Primárním účelem je jednoznačná definice operací, odpovědí, datových formátů, událostí a vlastností, které jsou vlastní připojeným zařízením (kamera, tiskárna, apod.). První čtyři nejvýznamější bity datacode definují typ, kategorii kódu. Také indikují, zda se jedná o standardní nebo specifický kód danému výrobci. Viz. tabulka 2. Na příkladu 1 je znázorněn datacode uzávěrky fotoaparátu, kdy první čtyři bity definují typ jako nastavení fotoaparátu a druhá část samotný kód.

Tabulka 1: Standardní datové typy

Jméno	Velikost (bajty)	Typ
OperationCode	2	Datacode (UINT16)
ResponseCode	2	Datacode (UINT16)
EventCode	2	Datacode (UINT16)3
DevicePropCode	2	Datacode (UINT16)
ObjectFormatCode	2	Datacode (UINT16)
StorageID	4	Special (UINT32)
ObjectHandle	4	Handle (UINT32)
DateTime	různá	String
DeviceInfo	různá	Dataset
StorageInfo	různá	Dataset
DevicePropDesc	různá	Dataset
DevicePropDescEnum	různá	enumerační forma DevicePropDesc
DevicePropDescRange	různá	rozsah z DevicePropDesc
Object	různá	různý

Příklad 1
Datacode uzávěrky fotoaparátu u značky Canon

$$\underbrace{1101}_{\text{typ}} - \underbrace{000000011110}_{\text{kód vlastnosti}}$$

■

Tabulka 2 zobrazuje standardem definované formátování datacode. Bit 15 nabývá hodnoty 1, pokud je kód specifický pro výrobce, tedy nedefinovaný standardem. Z tabulky je zřetelné, že kód uvedený v příkladu 1 je specifikován výrobcem a musí být interpretován podle VendorExtensionID a VendorExtensionVerison z datasetu Device info, které je blíže popsáno v části 2.1.2.3

2.1.2.2 Handly

Handly (Handles) jsou 32 bitové unsigned integery (UINT32), které musí být unikátní k danému zařízení a umožňují přístup k logickým nebo fyzickým prvkům připojeného zařízení. Handly typicky přetrvávají po dobu jedné relace 2.1.3, ale mohou přetrvávat i déle. Jedinou vlastností handlu je jeho unikátnost, na vlastní hodnotě nezáleží. Handly se používají k odkazování na objekty fotografií i dalších objektů, u kterých se očekává, že budou na vyžádání okamžitě k dispozici. Typicky se jedná o snímky nebo videa, ale může jím být i složka adresářového systému, která jako taková žádný objekt nevrací. Jejich obsah je pak odkazován jako ObjectHandles, tedy handl pro manipulaci s objekty. Z ObjectHandles lze získat ObjectInfo data, což jsou informace o daném objektu.

Tabulka 2: Formátování Datacode

Bit 15	Bit 14	Bit 13	Bit 12	Bit 11-0	Typ kódu
0	0	0	0	Libovolné	Nedefinováno (Nepoužívá se)
0	0	0	1	Libovolné	Standardní OperationCode
0	0	1	0	Libovolné	Standardní ResponseCode
0	0	1	1	Libovolné	Standardní ObjectFormatCode
0	1	0	0	Libovolné	Standardní EventCode
0	1	0	1	Libovolné	Standardní DevicePropCode
0	1	1	0	Libovolné	Rezervováno
0	1	1	1	Libovolné	Rezervováno
1	0	0	0	Libovolné	Nedefinováno
1	0	0	1	Libovolné	Specifický OperationCode
1	0	1	0	Libovolné	Specifický ResponseCode
1	0	1	1	Libovolné	Specifický ObjectFormatCode
1	1	0	0	Libovolné	Specifický EventCode
1	1	0	1	Libovolné	Specifický DevicePropCode
1	1	1	0	Libovolné	Rezervováno
1	1	1	1	Libovolné	Rezervováno

2.1.2.3 DeviceInfo Dataset

Tento dataset udržuje informace o zařízení. Iniciátor může informace získat od respondenta bez toho, aby byla otevřena relace. Udržuje informace jak o samotném zařízení, tak i o jeho schopnostech. Informace je statická. Schopnosti se nemohou během relace změnit. Z toho vyplývá, že DeviceInfo vždy reflektují možnosti zařízení v tu danou chvíli, v tom daném režimu. Je-li například fotoaparát v režimu natáčení videa, pak DeviceInfo vrátí schopnosti týkající se jen natáčení videa a ne schopnosti, které vykazuje režim fotografie, apod. Pokud se ale během relace režim zařízení změní, tak by měla být zavolána událost DeviceInfoChange, která reflektuje změny, které proběhly. Tabulka 3 zobrazuje obsah DeviceInfo DataSet.

Tabulka 3: DeviceInfo DataSet

Položka	Pořadí	Velikost (bajty)	Datový typ
StandartVersion	1	2	UINT16
VendorExtensionID	2	4	UINT32
VendorExtensionVersion	3	2	UINT16
VendorExtensionDesc	4	Různá	String
FunctionalMode	5	2	UINT16
OperationsSupported	6	Různá	OperationCode Array
EventsSupported	7	Různá	EventCode Array
DevicePropertiesSupported	8	Různá	DevicePropCode Array
CaptureFormats	9	Různá	ObjectFormatCode Array
ImageFormats	10	Různá	ObjectFormatCode Array
Manufacturer	11	Různá	String
Model	12	Různá	String
DeviceVersion	13	Různá	String
SerialNumber	14	Různá	String

StandartVersion: Popisuje nejvyšší verzi standardu, která je podporována zařízením. Je reprezentována ve stovkách. Tedy, je-li verze například 1.24, pak bude uložena jako 124.

VendorExtensionID: Obsahuje kontext k reprezentaci specifických funkcí, vlastností, které jsou definované výrobcem.

VendorExtensionVersion: Verze specifických rozšíření. Stejně jako StandartVersion je uložená ve stovkách.

VendorExtensionDesc: Volitelný řetězec k popisu VendorExtensionID. Jedná se pouze o informativní pole.

FunctionalMode: Volitelné pole. Ukládá informaci o režimu zařízení. Pokud má zařízení jen jeden režim, pak bude hodnota vždy 0. U digitálního fotoaparátu mohou hodnoty například být: Standardní mód, Režim spánku, Režim fotografie, apod.

OperationsSupported: Jedná se o pole obsahující kódy, reprezentující funkce, které zařízení v daném režimu podporuje.

EventsSupported: Pole, reprezentující události, které zařízení generuje v odpovídajících situacích. Závisí na OperationsSupported.

DevicePropertiesSupported: Pole kódů, které reprezentují vlastnosti zařízení. Je možné je číst a modifikovat.

CaptureFormats: List datových formátů, které dokáže zařízení vytvořit. Typicky se jedná o objekty fotografií nebo videí.

ImageFormats: Konkrétní formáty, které je možné pořizovat - JPEG, CR2, MOV, apod.

Manufacturer: Volitelné pole, které ukládá název výrobce Respondenta.

Model: Volitelné pole, ukládá název modelu zařízení Respondenta.

DeviceVersion: Volitelné pole, které obsahuje verzi firmwaru zařízení Respondenta.

SerialNumber: Volitelné pole. Unikátní číslo, které identifikuje zařízení.

2.1.2.4 StorageInfo Dataset

Tento dataset je použit k zaznamenání informací o úložném zařízení. Obsah této datové struktury je zobrazen v tabulce 4

Tabulka 4: StorageInfo DataSet

Položka	Pořadí	Velikost (bajty)	Datový typ
StorageType	1	2	UINT16
FilesystemType	2	2	UINT16
AccessCapability	3	2	UINT16
MaxCapacity	4	8	UINT64
FreeSpaceInBytes	5	8	UINT64
FreeSpaceInImages	6	4	UINT32
StorageDescription	7	Různá	String
VolumeLabel	8	Různá	String

StorageType: Identifikuje typ úložiště. Typicky se úložiště rozděluje na typy *ROM* nebo *RAM* a zda-li je pevné nebo odjímatelné.
FilesystemType: Volitelná položka. Zobrazuje typy souborových systémů v zařízení. Viz. tabulka 5

Tabulka 5: FilesystemType

Kód	Popis
0x0000	Nedefinován
0x0001	Plochý
0x0002	Hierarchický
0x0003	DCF
Vše ostatní s bitem 15 na 0	Rezervováno
Vše ostatní s bitem 15 na 1	Rezervováno

AccessCapability: Popisuje, zda úložiště umožňuje čtení a zápis nebo jen čtení. Dále lze definovat, zda lze položky pouze číst a nebo zda je možné objekty i mazat.
MaxCapacity: Volitelné pole. Zobrazuje celkovou kapacitu úložiště v bajtech.
FreeSpaceInBytes: Ukazuje dostupné místo v úložišti v bajtech.
FreeSpaceInImages: Počet snímků, které je možné uložit, než dojde místo v úložišti. Závisí na nastavení parametrů expozice.
StorageDescription: Volitelná položka, která umožňuje přidat popis úložiště.

VolumeLabel: Volitelné pole. Ukládá název úložiště.

2.1.3 Relace

Relace je logické spojení mezi dvěma zařízeními, mezi kterými je ObjectHandles a StorageID neměnné.

Není nutné, aby byla relace otevřená, když se získávají GetDeviceInfo informace, ale musí být otevřená pro přenos obrázků, datových objektů a jejich deskriptorů, jako například StorageInfo a ObjectInfo. Všechny takto získané informace jsou považovány za validní, pokud není explicitně vrácena chyba. Relace je poté otevřená až do chvíle, dokud není uzavřen komunikační kanál nebo není provedena operace CloseSession. Každá relace má své SessionID, které se skládá z unikátního 32 bitového unsigned integeru (UINT32). SessionID jsou přiděleny Iniciátorem jako parametr OpenSession operace a musí být nenulové.

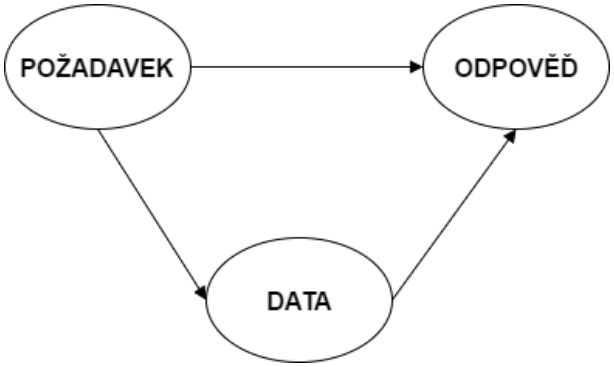
2.1.4 Transakce

Všechny transakce jsou považovány za atomické operace a za jejich vyvoláním by měl být jediný požadavek vyvolaný Iniciátorem nebo Respondentem. Standardně jsou transakce synchronní, blokující v rámci jedné relace. Pokud je zařízení schopné provozovat více relací zároveň, pak musí existovat asynchronně. Typickým příklad asynchronní operace u digitálních fotoaparátů je InitiateCapture. Jedná se o požadavek na vyfocení snímků bez potvrzení, že byl vyfocen a uložen na paměťové médium. Jediné potvrzení přichází v okamžiku, kdy byl požadavek zachycen fotoaparátem. V případě, že zařízení již vykonává nějakou asynchronní operaci a není schopné vykonat další, mělo by ohlásit, že je zaneprázdněné pomocí Device_Busy odpovědi. Iniciátor by se pak měl o stejnou operaci pokusit později.

Transakce se skládá ze tří fází. Fáze jsou následující:

- **Data** - v závislosti na prováděné operaci nemusí být přítomna. Data mohou být posílána buď z Iniciátora do Respondenta nebo z Respondenta do Iniciátora, ale nikdy v obou směrech u jediné operace.
- **Požadavek** - je vždy přítomný. Iniciátor nebo Respondent posílají požadavek na Data nebo na Odpověď.
- **Odpověď** - je vždy přítomný. Požadavek na Odpověď může přijít od Požadavku nebo od Data.

Jednotlivé fáze a směr toku informací ukazuje obrázek 2.



Obrázek 2: Sekvence transakcí

2.1.4.1 Operace Požadavek

Požadavek se skládá z přenosu 30 bajtového OperationRequest datasetu od Iniciátora k Respondentovi. Obsah OperationRequest zobrazuje následující tabulka 6.

Tabulka 6: OperationRequest

Položka	Velikost	Typ
OperationCode	2	UINT16
SessionID	4	UINT16
TransactionID	4	UINT16
Parameter1	4	Různé
Parameter2	4	Různé
Parameter3	4	Různé
Parameter4	4	Různé
Parameter5	4	Různé

- **OperationCode** - Kód, který určuje, jaká operace je vyvolávána. Více v 2.1.2.1.
- **SessionID** - Unikátní identifikátor relace. Více v 2.1.3.
- **TransactionID** - Identifikátor, který by měl být unikátní v rámci jedné relace. Pro každou následující transakci by se měl vždy zvětšit o jedničku.
- **Parameter** - Až pět parametrů, které jsou přenášeny v rámci vyvolané operace definované OperationCode. Pokud je parametr nepoužíván, měl by být nastaven na 0x00000000. Pokud parametr ukládá hodnotu menší než 32 bitů, pak by nejméně významné bity měly ukládat hodnotu parametru a nejvýznamnější bity by poté měly být nastaveny na nuly.

2.1.4.2 Operace Odpověď

Stejně jako operace Požadavku se i Odpověď skládá z 30 bajtového OperationResponse datasetu, který odesílá Respondent Iniciátorovi. Viz. tabulka 7.

Tabulka 7: OperationResponset

Položka	Velikost	Typ
ResponseCode	2	UINT16
SessionID	4	UINT16
TransactionID	4	UINT16
Parameter1	4	Speciální
Parameter2	4	Speciální
Parameter3	4	Speciální
Parameter4	4	Speciální
Parameter5	4	Speciální

2.1.4.3 Operace Data

Jedná se o nepovinnou fázi. Je používána k přenosu dat větších, než která dokáží přenést OperationRequest nebo OperationResponse. Typicky se může jednat o obrázek nebo video. Přes Data se dále posílají všechny objekty, které jsou označeny jako specifické pro výrobce, a to přesto, že by se velikostně vlezly do standardních OperationRequest, či OperationResponse. Posílání dat je vždy jednosměrné.

Formát, ve kterém jsou data posílána, je odvislý od obsahu, který je posílán a od ObjectFormat, který ukládá hodnotu ObjectFormatCode, viz. tabulka 1. Způsob, jakým jsou data posílána, např. rozdělení na malé části, posílání s hlavičkou, apod. je odvislé od transportní vrstvy a PTP standart jako takový, jej nedefinuje.

2.1.5 Pořadí operací

Pořadí operací je závislé na dané situaci a na zařízeních, která mezi sebou komunikují. Iniciátor určuje pořadí operací, zatímco Respondent musí na požadavky odpovídat. Odpovídat musí na standardní operace, jako nastavení fotoaparátu, pořízení snímku, apod., stejně jako na události. Iniciátor by měl být vždy připraven přijmout událost asynchronně vzhledem k datové fázi. Stejně tak Respondent musí být připraven přijímat události. Zde je několik příkladů typických scénářů. 0x00000000 značí, že je parametr nepoužit. 0xFFFFFFF pak značí samotná data, která jsou přenášena jako parametr k dané operaci.

2.1.5.1 Scénář 1

Iniciátor požaduje od Respondenta získat všechny obrázky, ignorovat náhledy obrázků, objekty které nejsou obrázky a asociace. Pořadí operací zobrazuje tabulka 8.

Tabulka 8: Scénář 1

Krok	Akce Iniciátora	Parametr 1	Parametr 2	Parametr 3	Akce Respondenta
1	GetDeviceInfo	0x00000000	0x00000000	0x00000000	Pošli DeviceInfo Dataset
2	OpenSession	SessionID	0x00000000	0x00000000	Vytvoř ObjectHandles a StorageID
3	GetObjectHandles	0xFFFFFFFF	0xFFFFFFFF	0x00000000	Pošli ObjectHandle pole
4	GetObjectInfo	ObjectHandle 1	0x00000000	0x00000000	Pošli ObjectInfo 1 Dataset
5	Opakuj krok 4 pro každý ObjectHandle	ObjectHandle n	0x00000000	0x00000000	Pošli ObjectInfo n Dataset
6	GetObject	ObjectHandle 1	0x00000000	0x00000000	Pošli ObjectInfo 1 Dataset
7	Opakuj krok 6 pro každý ObjectHandle	ObjectHandle n	0x00000000	0x00000000	Pošli ObjectInfo n Dataset
8	Close Session	0x00000000	0x00000000	0x00000000	

2.1.5.2 Scénář 2

Iniciátor požaduje od Respondenta získat všechny náhledy obrázků, bez dalších druhů objektů a asociací. Iniciátor také požaduje množinu objektů typu obrázek. Pořadí operací zobrazuje tabulka 9.

Tabulka 9: Scénář 2

Krok	Akce Iniciátora	Parametr 1	Parametr 2	Parametr 3	Akce Respondenta
1	GetDeviceInfo	0x00000000	0x00000000	0x00000000	Pošli DeviceInfo Dataset
2	OpenSession	SessionID	0x00000000	0x00000000	Vytvoř ObjectHandles a StorageID
3	GetObjectHandles	0xFFFFFFFF	0xFFFFFFFF	0x00000000	Pošli ObjectHandle pole
4	GetObjectInfo	ObjectHandle 1	0xFFFFFFFF	0x00000000	Pošli ObjectInfo 1 Dataset
5	Opakuj krok 4 pro každý ObjectHandle	ObjectHandle n	0x00000000	0x00000000	Pošli ObjectInfo n Dataset
6	GetThumb	ObjectHandle 1	0x00000000	0x00000000	Pošli Thumb 1 Data
7	Opakuj krok 6 pro každý ObjectHandle	ObjectHandle n	0x00000000	0x00000000	Pošli Thumb n Data
8	GetObject	ObjectHandle a	0x00000000	0x00000000	Pošli Thumb a Data
9	Opakuj krok 8 pro každý vybraný ObjectHandle	ObjectHandle m	0x00000000	0x00000000	Pošli Thumb m Data
10	CloseSession	0x00000000	0x00000000	0x00000000	Nic

2.1.5.3 Scénář 3

Iniciátor posílá všechny objekty Respondentovi, ten si sám určuje, kde přijaté objekty umístí. Pořadí operací zobrazuje tabulka 10.

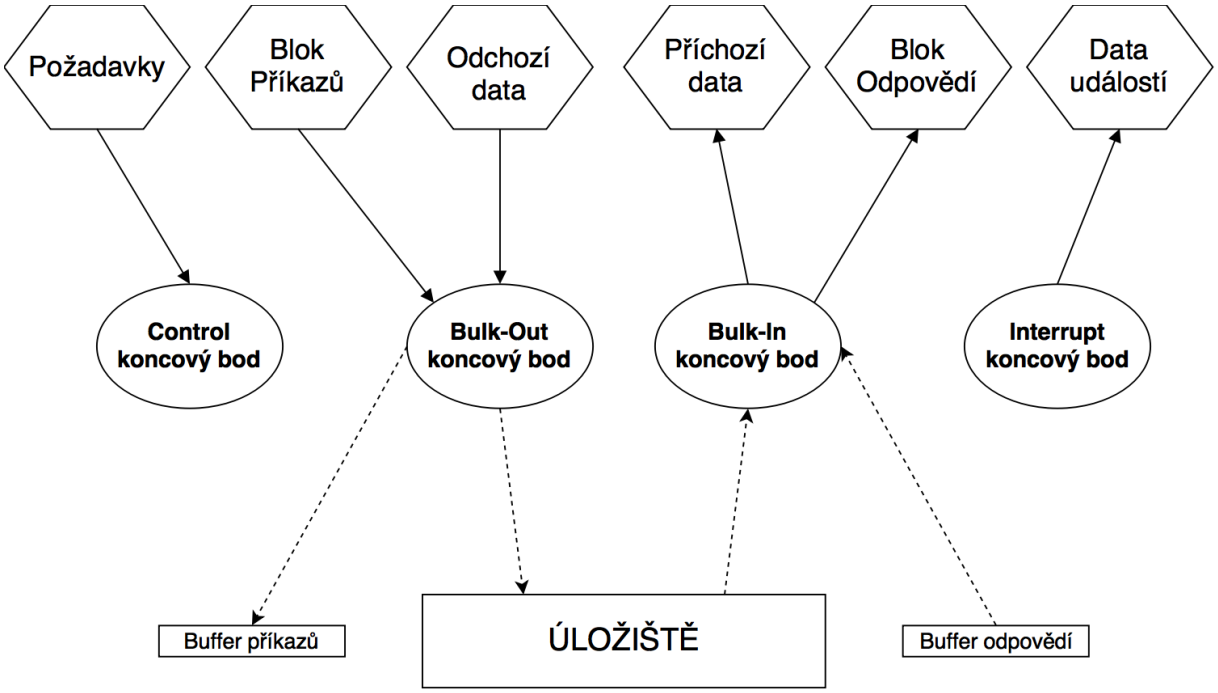
Tabulka 10: Scénář 3

Krok	Akce Iniciátora	Parametr 1	Parametr 2	Parametr 3	Akce Respondenta
1	GetDeviceInfo	0x00000000	0x00000000	0x00000000	Pošli DeviceInfo Dataset
2	OpenSession	SessionID	0x00000000	0x00000000	Vytvoř ObjectHandles a StorageID
3	SendObjectInfo	0x00000000	0x00000000	0x00000000	Alokuj pamět, přiděl nový ObjectHandle, vrať StorageID, rodičovský ObjectHandle a ObjectHandle
4	SendObject	0x00000000	0x00000000	0x00000000	Zapiš data do úložiště, potvrď odesláním SendObjectInfo
5	Opakuj krok 3 pro každý objekt				
6	CloseSession	0x00000000	0x00000000	0x00000000	Nic

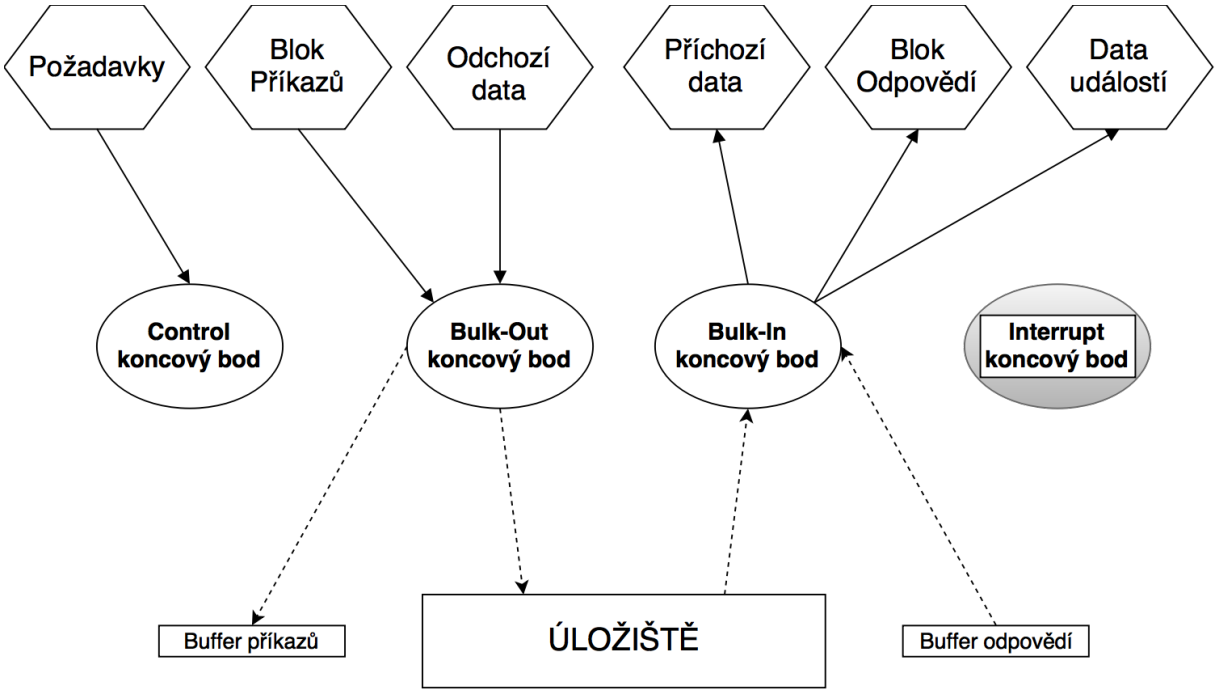
2.1.6 Výrobcem specifikované operace

PTP standart jako takový umožňuje výrobcem specifikované operace bez toho, aby byl porušen standard. Toto je blíže popsáno v 2.1.2.3. Výrobci fotoaparátů ale ne vždy standard dodržují. Příkladem je značka Canon, jak bude uvedeno níže. Na obrázku 3 je zobrazen tok dat a informací u fotoaparátu ze standardních koncových bodů, definovaných standardem.

Výrobce Canon u svých fotoaparátu EOS provedl změnu, kdy události posílá místo na standardem definovaný interrupt koncový bod na bulk-In koncový bod, který se normálně používá jen pro větší objemy dat, jak ukazuje obrázek 4. Tyto změny znesnadňují implementace aplikací, komunikujících s fotoaparáty Canon, neboť sám výrobce je nikde nepopisuje.



Obrázek 3: Tok informací u digitální kamery dle standartu



Obrázek 4: Tok informací u digitální kamery dle výrobce Canon

2.2 Implementované funkce

Následující funkce využívají knihovnu libgphoto2 [3], která implementuje základy standartu PTP a základy specifických odlišností výrobců fotoaparátů.

2.2.1 InitCamera

Jedná se o funkci, která nejprve vytvoří nový kontext, ten umožňuje zpracovávat zpětná volání, předávat chybové hlášky, informaci o stavu, přerušení stahování objektů, apod. Obvykle je kontext předáván jako parametr funkce. Jako druhý krok je provedena alokace paměti pro novou kameru.

Poslední krok poté detekuje připojené zařízení a provede inicializaci na základě daných parametrů kontextu a alokované paměti. V té chvíli je fotoaparát obsazen a nemůže k němu přistupovat žádné jiné zařízení. Také se získají informace o schopnostech kamery a nastavitelných parametrech, jakož i aktuální režim, ve kterém se kamera nachází. Další důležitou informací, která je z kamery získána, je StorageID 1, díky kterému bude možné dále přistupovat k dostupným úložným zařízením.

2.2.2 ExitCamera

Uvolní paměť, která byla alokována funkcí *InitCamera*, popřípadě paměť, která byla alokována jinými funkcemi při běžné komunikaci s kamerou. Také uvolní komunikační kanál pro možnost nové inicializace nebo připojení z jiného zařízení. Pokud je kamera momentálně zaneprázdněna, například vytváří snímek, stahuje objekt, apod., vrátí se chybová hláška a zařízení nebude možno uvolnit. Po dokončení probíhající akce je pro uvolnění nutné funkci zavolat znovu.

2.2.3 GetCameraStatus

Tato funkce běží v samostatném vlákně a v pravidelných intervalech kontroluje, zda-li je kamera korektně připojena. V případě že není, tak řídící software panoramatické hlavy provede reinicializaci. Funkce mění hodnotu boolovské proměnné *isCameraConnected*, která určuje momentální stav připojení kamery. Tu může kromě vlákna na kontrolu stavu kamery měnit i hlavní vlákno, byl proto využit synchronizační prostředek mutex, aby nedošlo k přístupu do kritické sekce oběma vlákny zároveň a hodnota proměnné byla korektní. Kontrola stavu je dále implicitně volána při požadavku na změnu nastavení kamery nebo při připojení klienta pro vzdálené ovládání.

2.2.4 GetFreeMemory

Funkce vrací *char array* s informací o volné paměti na připojeném úložišti. Hodnota je vrácena v kilobajtech. Brána je v potaz pouze volná kapacita externího úložiště kamery, typicky paměťové karty, přestože by bylo možné získat i informace o volné vnitřní paměti nebo paměti RAM.

Tyto hodnoty ovšem nejsou pro řídící software panoramatické hlavy relevantní a jsou proto ignorovány.

2.2.5 CaptureImage

Funkce, která je klíčová pro rozšíření řídícího software motorizované panoramatické hlavy. V základu pořizuje snímek, uloží ho na externí paměťové médium a vrátí potvrzení o úspěšném nebo neúspěšném vytvoření.

Klíčovou vlastností této funkce je právě potvrzení, zda-li byl snímek úspěšně vytvořen a uložen. Předchozí řídící software neumožňoval aktivní spojení s kamerou a "nevěděl", zda byl snímek opravdu vytvořen a uložen. Praxe ukázala, že je to nesmírně důležité, neboť se stávalo, že kamera v důsledku zaplnění vnitřního bufferu, nedostatku paměti nebo například při nemožnosti zaostřit scénu, snímek nevytvořila, čímž několikahodinové focení panoramata, které se mohlo skládat z více než tisíce snímků, znehodnotila. Průběh funkce je následující:

- **Vyfocení** - odešle se požadavek na vyfocení snímku, kde je jako parametr definována připojená kamera, typ objektu, který chceme pořizovat. To může být obrázek, video nebo například zvukový záznam. Dále může být definován název a cesta k souboru na paměťovém médiu. Posledním parametrem je poté kontext, vzniklý při *InitCamera* 2.2.1.
- **Uložení** - vyfocený snímek je uložen na paměťovém médiu. Cesta je definována ve struktuře *CameraFilePath*, která má dvě položky: *name* - jméno objektu a *folder* - cesta k souboru. Pokud tyto položky nejsou implicitně nastaveny, bude použito defalutní nastavení kamery.
- **Návratová hodnota** - Funkce po dokončení každé výše popsané fáze potvrzuje, zda-li byla akce úspěšná. V případě ukládání také vrátí cestu k uloženému souboru.

Implementace samotné funkce je následující:

```
int CaptureImage()
{
   int ret; // return value
   CameraFile *file; //object file
   CameraFilePath camera_file_path;

   DEBUG_PRINT("CAMERA(Capture): Capturing.\n");

   //capture image and save to given path
   ret = gp_camera_capture(camera, GP_CAPTURE_IMAGE, &camera_file_path, context
       );
```

```
    if (ret < GP_OK) {
        DEBUG_PRINT("CAMERA(Capture): Camera failed to capture image.\n");
        gp_camera_free(camera); // unref camera,
        return 0;
    }

    DEBUG_PRINT("CAMERA(Capture): Retval: %d\n", ret);
    DEBUG_PRINT("CAMERA(Capture): Pathname on the camera: %s/%s\n",
        camera_file_path.folder, camera_file_path.name);
}
```

Výpis 1: Vyfocení snímku s potvrzením

2.2.6 TriggerImage

Na rozdíl od *CaptureImage*, tato funkce nečeká na potvrzení o uložení snímku. Vyšle požadavek na pořízení snímku, kamera musí odpovědět, zda je schopna a připravena snímek pořídit. Vrátí chybu v případech, kdy kamera není inicializována, připojena, je zaneprázdněna nebo není v režimu, kdy je možné snímek pořídit.

Funkce je výhodná v případech, kdy je potřeba vyfotit snímky ve velké rychlosti za sebou. U funkce *CaptureImage* je ve většině případů časově nejnáročnějším úkonem ukládání snímků do úložiště, a tuto dobu, než je snímek uložen, nelze v některých případech použití tolerovat a čekat na dokončení. Není zde absolutní jistota pořízení snímku, ale je tu jistota, zda je kamera připravena a o vyfocení snímku se pokusí. Praktické testy ukázaly, že používání je spolehlivé.

2.2.7 CapturePreview

Vytvoří náhledovou fotografii právě snímané scény. Funkci využívá vzdálené ovládání, kdy je možné na dálku po vyžádání pořídit snímek, který je poté zobrazen v klientském prohlížeči. Velikost náhledu je přímo odvislá od rozlišení displeje fotoaparátu, neboť je náhled odebrán z videosignálu, který je posílán na displej fotoaparátu při tzv. Live-View, což je režim fotoaparátu, kdy je obraz snímaný objektivem živě přenášen na displej fotoaparátu. Je-li fotoaparát v tomto režimu, tak při klasickém pořízení snímku je automaticky sejmut i náhled, který je uložen v datech vedle snímku v plném rozlišení.

Pro pořízení náhledu je proto nejdříve nutné fotoaparát přepnout do Live-View režimu, poté je pořízen snímek v plném rozlišení, který se uloží do operační paměti, z tohoto snímku se odebere pouze náhled, ten se přenese do klienta a v kameře je zahozen. Náhled je uložen do definovaného adresáře ve formátu jpeg, kdy je velikost v průměru 15KB. Přenos takového snímku do klienta je proto rychlý a nevytěžuje přenosové médium.

2.2.8 GetValuesRange

Bez implementace této funkce by nebylo možné spolehlivě nastavovat funkce připojeného fotoaparátu. Každý fotoaparát se liší v parametrech, které je u něj možné nastavit. Typickým příkladem je hodnota citlivosti ISO, kdy dražší fotoaparáty obvykle zvládají vyšší hodnoty než ty levnější. Druhým příkladem může být nastavení clony. Ta je přímo závislá na připojeném objektivu, a tedy jeho výměnou se změní i možné hodnoty clony. Aby bylo možné měnit parametry fotoaparátu z připojeného klienta, což může být mobil, počítač, nebo řídící jednotka panoramatické hlavy, je nutné nejdříve získat rozsah parametrů, které je možné nastavit. Funkce přijímá dva parametry. Prvním je *const char *name*. Ten definuje funkci, o které chceme získat rozsah nastavitelných hodnot. Taková funkce je nazývána widget. Parametr je přijímán jako pole charů, které je dále převedeno do datového kódu (datacodes) 2.1.2.1.

Do druhého parametru *char **res* jsou poté uloženy získané hodnoty. Bylo zvoleno dvourozměrné pole, aby bylo co nejjednodušší s nimi manipulovat z prostředí uživatelského rozhraní. Kvůli univerzálnosti jsou také všechny hodnoty, bez ohledu na typ, uloženy jako pole charů.

Samotná funkce poté vrací hodnotu nula, pokud funkce proběhla v pořádku. Implementace vypadá následovně:

```
int GetValuesRange(const char *name, char **res)
{
   const char *label;
   CameraWidgetType type;
   int ret;

   CameraWidget *widget = NULL, *child = NULL;

   ret = gp_camera_get_config(camera, &widget, context); // get all possible
       widgets
   if (ret < GP_OK) {
      DEBUG_PRINT("CAMERA(Values): camera_get_config failed: %d\n", ret);
      return ret;
   }
   ret = _lookup_widget(widget, name, &child); // find widget by name parameter
   if (ret < GP_OK) {
      DEBUG_PRINT("CAMERA(Values): lookup widget failed: %d\n", ret);
      return 0;
   }

   ret = gp_widget_get_type(child, &type); // what type is it? Can be set to
       value?
```

```
    if (ret != GP_OK)
        return ret;
    ret = gp_widget_get_label(child, &label); // get label of found widget
    if (ret != GP_OK)
        return ret;

    int cnt, i;
    char *current;

    ret = gp_widget_get_value(child, &current); // get current setting of widget
    if (ret == GP_OK) {
        cnt = gp_widget_count_choices(child); // how many parameters are there

        for (i = 0; i < cnt; i++) {
            const char *choice;
            ret = gp_widget_get_choice(child, i, &choice); // save all to array
            res[i] = (char *)choice;
        }
    }
    else {
        gp_context_error(context, ("Failed to retrieve values of radio widget %s.
            "), name);
    }
    return GP_OK;
}
```

Výpis 2: Získání všech nastavitelných parametrů fotoaparátu

2.2.9 GetCameraName, GetLensName

Funkce vrací oficiální názvy připojeného fotoaparátu a kamery. U některých fotoaparátů vrací v názvu i aktuální hodnotu firmware. U objektivů je kromě názvu vrácena i vývojová generace objektivu. Jaké hodnoty budou v názvu kamery a objektivu obsaženy, nejsou definované žádným psaným standardem a jsou zcela na výrobci.

2.2.10 GetShutterSpeed a SetShutterSpeed

Umožňuje získat aktuální nastavenou hodnotu závěrky. *SetShutterSpeed* umožňuje hodnotu nastavit. Hodnota je funkci předávána jako parametr.

Vnitřní logika fotoaparátu kontroluje přijímaný parametr, v případě předání špatného parametru by proto nemělo hrozit riziko závažné chyby fotoaparátu.

2.2.11 GetAperture a SetAperture

Umožňuje získat aktuální nastavenou hodnotu clony. Pomocí *SetAperture* lze hodnotu nastavit.

Je nutné, aby připojený fotoaparát umožňoval elektronické nastavení clony a fotoaparát objektiv rozpoznal. Není možné přes PTP rozhraní spojení mezi kamerou a objektivem nijak ovlivnit. Správné rozpoznání objektivu lze ovšem ověřit funkcí *GetLensName*. Bohužel praxe ukázala, že s rozpoznáváním některých objektivů mohou být problémy. Typicky se to stává u velmi starých objektivů, řádově 10 a více let a zcela nových objektivů. Vždy záleží na spolupráci mezi výrobci fotoaparátů a objektivů tak, aby po vydání nového objektivu byl vydán i nový firmware kamery, který podporu daného objektivu obsahuje. U značky Canon tento problém nastal, když sám tento výrobce začal nabízet objektivy s novým pohonem STM. Většina fotoaparátů měla velké problémy s těmito objektivy a bohužel výrobce vydal novou verzi firmware jen pro své novější fotoaparáty, a tak majitelé starších fotoaparátů nemohli naplno nové objektivy využít.

2.2.12 GetISO a SetISO

Tyto funkce umožňují získat aktuální hodnotu citlivosti a také hodnotu nastavit. Hodnoty nastavení jsou čistě odvislé od možností připojeného fotoaparátu a omezeny hardwarovými parametry.

2.2.13 HDRCapture

HDR je zkratka pro High Dynamic Range. V oboru fotografie se jedná o takový snímek, který pokrývá vyšší dynamický rozsah, než běžná fotografie. Je vhodné ho použít v případech, kde je velký kontrast mezi světlými a tmavými místy snímané scény. Příkladem může být místnost s okny, kdy při pořízení klasické fotografie bude výsledek takový, že buď bude korektně exponovaná místnost a okno bude jasově přepálené nebo opačně bude místnost příliš tmavá a okno korektně nasnímané.

Správné expozice lze dosáhnout tak, že se vyfotí daná scéna s několika různými expozicemi. V případě výše popsaného příkladu se tak vyfotí snímek se správnou expozicí okna a další se správnou expozicí místnosti. Specializovaný software nebo samotný fotoaparát pak tyto snímky složí do jediného snímku s vyváženou expozicí. Při pořizování je důležité, aby všechny snímky byly foceny ze stejného místa. Ideální je použití fotografického stativu.

U panoramatické fotografie je výhodné, aby všechny expozice byly vyfoceny do samostatných RAW souborů, neboť je v závislosti na snímané scéně nutné dělat manuální úpravy expozice tak, aby po složení všech dílčích snímků panoramata byla expozice jednotná a panorama působilo dojmem jednolitého snímku. Většina fotoaparátů v dnešní době sice nabízí automatický režim

tvorby HDR, ale ve většině případů umožňují uložení pouze do jpeg souboru a možnost nastavení expozičních parametrů před vyfocením snímku je malé.

Byla proto implementována funkce *HDRCapture*, která umožňuje vyfotografovat scénu s různou expozicí. Konkrétně jsou při běhu funkce pořízeny v krátkém časovém úseku tři samostatné snímky. Funkce jako parametr přijímá hodnotu od 0 do 12, která reprezentuje o jak velké množství bude expozice posunuta níže a výše oproti správné expozici. V praxi je zavedena jednotka EV, tzv. expoziční hodnota. 0 EV je relativní hodnota, která udává optimální expozici pro danou scénu. Může to být expozice, která je nastavena automatikou fotoaparátu, či manuálně nastavena uživatelem. -1 EV značí, že oproti relativní hodnotě 0 EV bude na snímač fotoaparátu při pořízení snímku vpuštěno poloviční množství světla, než při 0 EV. Opačně pak při +1 EV bude vpuštěno dvojnásobně více světla.

Množství světla dopadající na snímač fotoaparátu je možné regulovat úpravou tří základních parametrů: expozičním časem, clonou a citlivostí ISO. Jakým způsobem budou parametry upraveny, je odvislé od samotného fotoaparátu a jeho algoritmů pro výpočet správné expozice, tak i od režimu, ve kterém se nachází. Bude-li například v režimu priority clony, bude primárně expozice upravena změnou clonového čísla. Pokud je fotoaparát v režimu manuál, je typicky měněna hodnota času závěrky. Následující tabulka 11 ukazuje možné parametry funkce a jejich odpovídající posun v jednotce EV:

Tabulka 11: Přijímaný parametr HDR funkce a odpovídající EV

Paramter (UINT32)	**hodnota EV**
0	vypnuto
1	+/- 1/3
2	+/- 1/2
3	+/- 2/3
4	+/- 1
5	+/- 1 1/3
6	+/- 1 1/2
7	+/- 1 2/3
8	+/- 2
9	+/- 2 1/3
10	+/- 2 1/2
11	+/- 2 2/3
12	+/- 3

2.2.14 GetHDRStatus

Získá hodnotu stavu HDR. Je-li zapnutá, získá aktuální nastavení.

2.2.15 GetBatteryLevel

Vrací aktuální hodnotu stavu baterie fotoaparátu. Hodnota je uvedena v procentech. Přečtena pak může být na displeji řídící jednotky nebo z klientského prohlížeče při vzdáleném ovládání.

2.2.16 GetAvailableShots

Funkce vrátí počet snímků, které je ještě možné pořídit v závislosti na volné paměti fotoaparátu a současném nastavení. Samotný algoritmus, který hodnotu počítá, je záležitostí fotoaparátu a jedná se o stejnou hodnotu, kterou je možné přečíst na displeji fotoaparátu. V případě, že je před focením panoramatického snímků hodnota dostupných snímků menší, než kolik jich má být celkově vyfoceno, pak je na displeji řídící jednotky vypsáno upozornění.

2.2.17 FocusStacking

Focus Stacking je technika fotografování, kdy je scéna pořízena několika snímky, a to tak, že každý snímek má jinou míru zaostření. Složením takových snímků do jediného lze poté dosáhnout ostrosti všech objektů dané scény. U gigapixelových snímků je míra zaostření velice důležitá a zároveň velkým problémem.

U použití objektivů s velkým ohniskem je prostor, kde jsou objekty korektně zaostřeny, úzký. Při focení se obvykle ostří na nekonečno, aby co největší část scény byla zaostřena. To má ovšem za následek, že objekty v popředí panoramata budou vždy rozostřené. U některých scén to nemusí být velký problém, např. jsou li v popředí jen vrcholky stromů, apod. Téměř vždy ale problém nastane, je-li focena městská zástavba. Neostré popředí snímku kazí celkový dojem a autor panoramata se tak připravuje o podstatnou část informací, které chtěl zachytit.

Proto byla implementována funkce *FocusStacking*, která naplno využívá možnosti aktivního rozhraní mezi fotoaparátem a panoramatickou hlavou. Princip fungováni je následující:

- **Nastavení ostřícího bodu 1** - Uživatel je pobízen, aby korektně zaostřil na první část scény. Není podstatné, zda se jedná o popředí nebo jinou část scény. Jakmile má korektně zaostřeno, pomocí tlačítka volbu potvrdí.
- **Sejmutí aktuálního zaostření 1** - Z kamery je sejmuta hodnota aktuálního zaostření. Jedná se o číselnou hodnotu v rozmezí od -32768 do 32768. Ta je uložena do proměnné.
- **Nastavení ostřícího bodu 2** - Uživatel je pobídnut, aby zaostřil druhou část snímku. Jakmile má hotovo, volbu musí potvrdit.
- **Sejmutí aktuálního zaostření 2** - Z kamery je sejmuta hodnota aktuálního zaostření. Jedná se o číselnou hodnotu v rozmezí od -32768 do 32768. Ta je uložena do proměnné.
- **Pořízení snímku** - Při samotném focení panoramata je před pořízením snímku vždy zaostřeno nejdříve na bod 1, pořízen snímek, zaostřeno na bod 2 a pořízen snímek. Celkový počet snímků je tedy dvojnásobný.

V době psaní této práce je funkce FocusStacking bezproblémově funkční u fotoaparátů značky Nikon. Fotoaparáty Canon a ostatních výrobců v tuto chvíli podporovány nejsou. Je to z toho důvodu, že starší modely neumožňují přenášet přes PTP protokol aktuální hodnotu zaostření a ani zaostření s dostatečnou přesností nastavit.

2.3 Uživatelské rozhraní

Uživatel má možnost ovládat panoramatickou hlavu pomocí šesti tlačítek a displeje, které jsou umístěný na hlavní jednotce připojené k motorizované hlavě. K již implementovaným funkcím z minulé verze software přibyly možnosti nastavení připojené kamery a dalších pokročilých funkcí jako je HDR nebo Focus Stacking.

2.3.1 Hlavní menu

Hlavní menu je automaticky zobrazeno po spuštění přístroje na displeji zařízení. Hlavní menu zobrazuje obrázek 5.

Obrázek 5: Ukázka obrazovky hlavního menu

2.3.2 Nové Panorama

Parametry nastavené v tomto podmenu jsou klíčové z hlediska focení panoramatu. Určují, kolik snímků bude vyfoceno, jak velký bude úhel záběru a jak rychle se všechny potřebné úkony dokončí.

2.3.3 Nastavení úhlu pohybem

Aby bylo možné co nejpřesněji nastavit úhly záběru, které definují velikost snímané scény, je možné pohybem panoramatické hlavy tyto hodnoty získat.

Po spuštění funkce je uživatel vyzván k postupnému vykonání tři kroků:

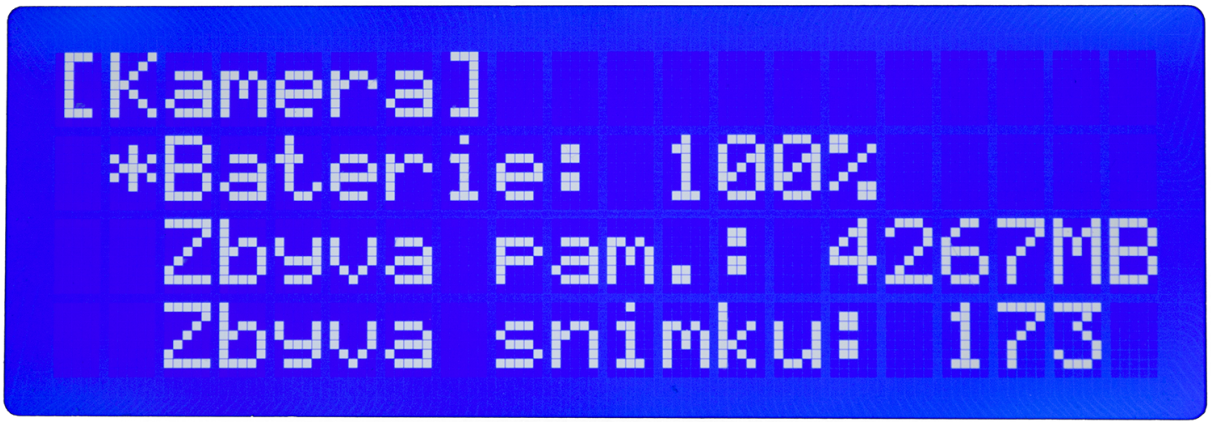
1. Nastavit panoramatickou hlavu pomocí stisknutí příslušných šipek do pozice, která odpovídá pravému hornímu rohu budoucího panoramatického snímku. Uložení pozice provede stisknutím potvrzovacího tlačítka.

2. Pohybem hlavy změní pozici na pravý dolní roh a volbu potvrdí stiskem tlačítka. V tomto okamžiku je již znám vertikální úhel a uživatel jeho hodnotu může zkontrolovat v informačním poli.

3. Posledním krokem je poté nastavení hlavy do pozice levého dolního rohu a potvrzení příslušným tlačítkem. V tomto okamžiku je znám i celkový horizontální úhel.

2.3.4 Kamera

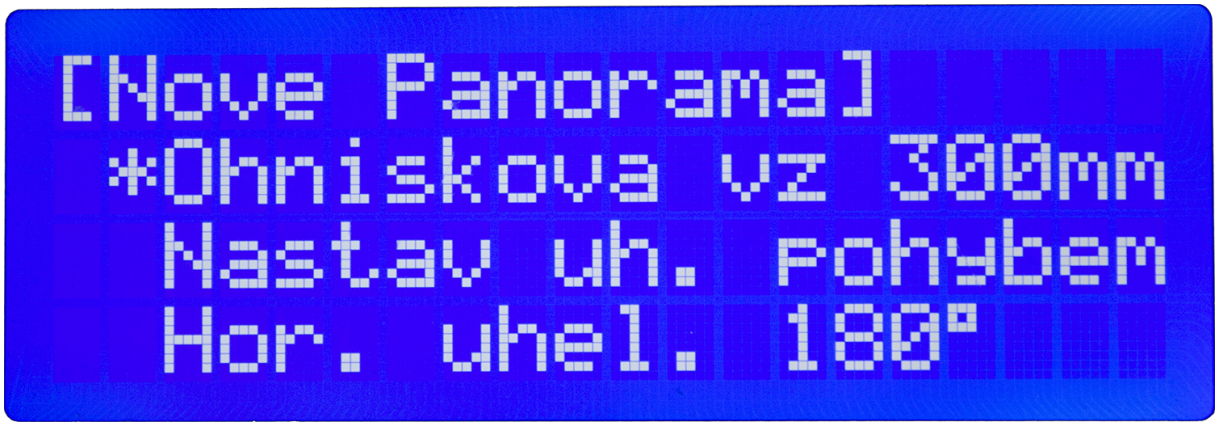
Je-li připojena kamera a spojení s řídící jednotkou je korektně ustaveno, je zde možné získat základní informace o připojeném fotoaparátu a objektivu, jako i informaci o stavu baterie. Dále je umožněno nastavit základní parametry expozice. Viz. obrázek 6.



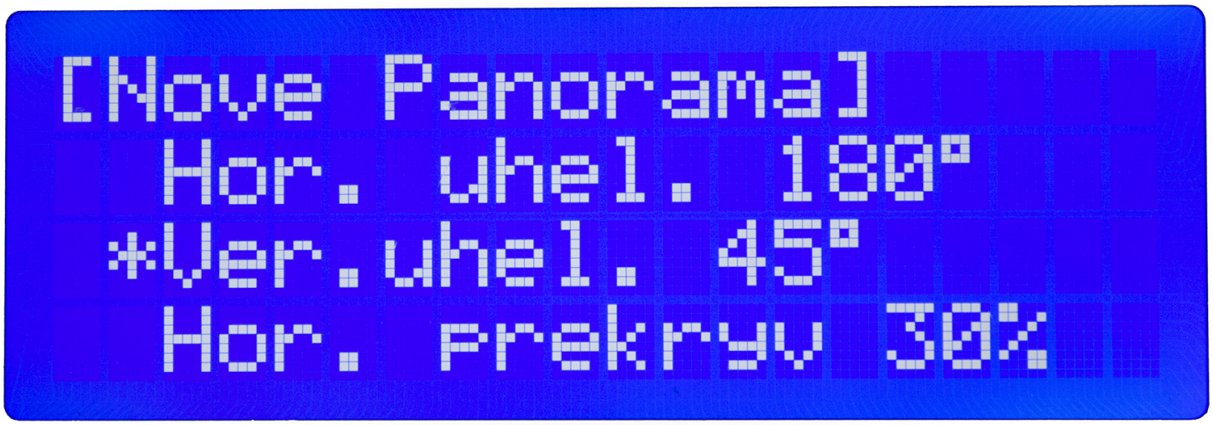
Obrázek 6: Ukázka obrazovky kamera

2.3.5 Pan Info

Zde jsou zobrazeny informace o panoramatickém snímku. Hodnoty jednotlivých položek jsou přímo závislé na parametrech nastavených uživatelem v podmenu *Nove panorama*. Viz. obrázek 7 a 8.



Obrázek 7: Ukázka obrazovky nastavení panoramata 1



Obrázek 8: Ukázka obrazovky nastavení panoramata 2

3 Dálkové ovládání

Druhou významnou částí rozšíření je umožnění ovládání motorizované panoramatické hlavy a připojeného fotoaparátu na dálku. Praktické používání předchozí verze ukázala, že je toto rozšíření potřebné a užitnou hodnotu systému výrazně zvýší. Ve velmi častých případech je pro pořízení co nejlepšího panoramata nutné upevnění motorizované hlavy do těžko přístupných míst, kde je manipulace velice obtížná. Pořizování samotných snímků poté může trvat i několik hodin a setrvávat na těchto těžko dostupných místech po celou dobu focení, je velice obtížné a mnohdy i nebezpečné. Díky vzdálenému ovládání stačí celý systém pouze upevnit na zvolené místo a nastavení i focení řídit pohodlně na bezpečném místě.

Dle zadání měla být vyvinuta aplikace pro mobilní operační systém Android. Po konzultaci s vedoucím diplomové práce ale bylo dohodnuto, že mnohem lepším řešením bude vývoj takové aplikace, která umožní vzdálené ovládání z běžného internetového prohlížeče. Díky tomu se počet zařízení, kde bude možné hlavu ovládat, mnohonásobně zvýší.

3.1 WebSocket

Jako prostředek pro implementaci řídící aplikace vzdáleného ovládání byl zvolen komunikační protokol WebSocket. Hlavní výhody tohoto protokolu jsou následující:

- **Obousměrné spojení** - Websocket umožňuje obousměrnou komunikaci v rámci jednoho TCP spojení. Stále používaný způsob komunikace mezi serverem a klientem bez použití Websocket se zakládá na tzv. dotazování (polling), kdy v případě, že bylo potřeba získat informace ze serveru, bylo nutné odeslat nový požadavek od klienta. Informace vždy proudily maximálně jedním směrem, nikdy oběma zároveň. Každá odeslaná zpráva pak musela obsahovat obsáhlou HTTP hlavičku, mnohdy větší, než velikost samotné zprávy, což zbytečně zatěžovalo server. Více v 3.1.2
- **Jediné TCP spojení** - Pro obousměrné spojení s klientem je zapotřebí jediného TCP spojení oproti několika, která jsou potřebná u dotazování (polling).
- **Latence** - Vzhledem k tomu, že spojení je obousměrné, pro každou odpověď od serveru nemusí být posílán požadavek od klienta a zprávy neobsahují celou HTTP hlavičku, je latence výrazně nižší, což v případě aplikací s častou aktualizací dat hraje významnou roli. Více v kapitole 3.1.3

3.1.1 Handshake

Aby se mohlo utvořit spojení mezi klientem a serverem, je potřeba poslat handshake. Klient i server používají každý svou verzi handshake. Z důvodu zachování kompatibility s HTTP je definován jako HTTP Upgrade request. Příklad, jak může vypadat, je následující:

Příklad 2
Ukázka klientkého handshake websocket
GET /remotecontrol HTTP/1.1
Host: www.server.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Origin: http://www.klient.com
Sec-WebSocket-Protocol: remote control
Sec-WebSocket-Version: 13

■

- **GET /remotecontrol HTTP/1.1** - */remotecontrol* se používá k identifikaci koncového bodu WebSocket spojení, aby bylo možné využívat několik domén zároveň z jedné IP adresy, a také aby jediný WebSocket server mohl obsluhovat vícero koncových bodů.
- **Host: www.server.com** - určuje adresu serveru, aby si klient i server mohli ověřit, že se shodli na stejné adrese serveru.
- **Upgrade: websocket** - žádost o povýšení ze stávajícího protokolu na WebSocket
- **Connection: Upgrade** - typ požadavku
- **Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==** - náhodně zvolená 16 bajtová hodnota, která musí být kódována v base64. Používá se k ověření, že server handshake v pořádku přijal.
- **Origin: http://www.klient.com** - toto pole je povinné v případě, že požadavky přichází z prohlížeče klienta. V případě, že klient prohlížeč nepoužívá, není povinné. Definuje původ klientského kódu. Například, pokud se klientský kód stažený z adresy www.klient.com snaží vytvořit spojení s www.server.com, pak hodnota Origin pole bude www.klient.com
- **Sec-WebSocket-Protocol: remote** - nepovinné pole, které určuje, jaké pod-protokoly jsou akceptovatelné pro klienta. Server poté může, ale i nemusí jeden vybrat a oznámí to klientovi.
- **Sec-WebSocket-Version: 13** - verze protokolu.

Handshake serveru je obsahově chudší a může například vypadat takto:

Příklad 3
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade

Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=

■

- **HTTP/1.1 101 Switching Protocols** - Status */101* určuje, že handshake proběhl v pořádku. Jakýkoli jiný kód značí opak.
- **Upgrade: websocket** - žádost o povýšení ze stávajícího protokolu na WebSocket.
- **Connection: Upgrade** - typ požadavku.
- **Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=** - Hodnota je vytvořena způsobem, kdy je vzat z klientského handshake Sec-WebSocket-Key a spojen s GUID (Globally Unique Identifier), který má hodnotu 258EAFA5-E914-47DA-95CA-C5AB0DC85B11. Výsledný klíč je poté zakódován SHA-1 algoritmem a je vrácen v handshake serveru. Klíč slouží k potvrzení úspěšného handshake.

3.1.2 Vytíženost spojení

Hlavní výhodou WebSocket oproti dotazovací (polling) metodě je nižší vytížení spojení mezi klientem a serverem. Díky tomu je tak možné provozovat aplikace s velkým počtem klientů, kteří vyžadují časté informace ze serveru s mnohem menšími nároky na serverovou infrastrukturu.

Příklad níže ukazuje porovnání vytíženosti spojení mezi dotazovací metodou a WebSocket. V příkladu se počítá s tím, že HTML hlavička, která musí být při dotazovací metodě pokaždé odeslána jak při HTTP request, tak i HTTP response a data, měli v průměru kolem 871 bajtů. U WebSocket se hlavičky odesílají pouze při navázání spojení. Datové rámce poté zabírají pouze 2 bajty.

Dotazování

- **1000 klientů se dotazuje serveru každou sekundu** - Datový tok je 871 x 1000 = 871000 bajtů = 6968000 bitů/s = 6.6 Mbit/s.
- **10000 klientů se dotazuje serveru každou sekundu** - Datový tok je 871 x 10000 = 8710000 bajtů = 69680000 bitů/s = 66 Mbit/s.
- **100000 klientů se dotazuje serveru každou sekundu** - Datový tok je 871 x 100000 = 87100000 bajtů = 696800000 bitů/s = 665 Mbit/s.

WebSocket

- **1000 klientů přijímá zprávu každou sekundu** - Datový tok je 2 x 1000 = 2000 bajtů = 16000 bitů/s = 0.015 Mbit/s.
- **10000 klientů přijímá zprávu každou sekundu** - Datový tok je 2 x 10000 = 20000 bajtů = 160000 bitů/s = 0.153 Mbit/s.

- **100000 klientů přijímá zprávu každou sekundu** - Datový tok je 2 x 100000 = 200000 bajtů = 1600000 bitů/s = 1.526 Mbit/s.

3.1.3 Latence

Vedle nižší vytíženosti síťového spojení je latence druhým problémem, kvůli kterému standard WebSocket vznikl. Tabulka 12 ukazuje porovnání latence mezi dotazovací metodou a WebSocket. V příkladu se počítá s tím, že odeslání jednoho požadavku trvá 50ms, stejnou dobu trvá i příjem požadavku.

Tabulka 12: Porovnání latence mezi dotazování (Polling) a WebSocket

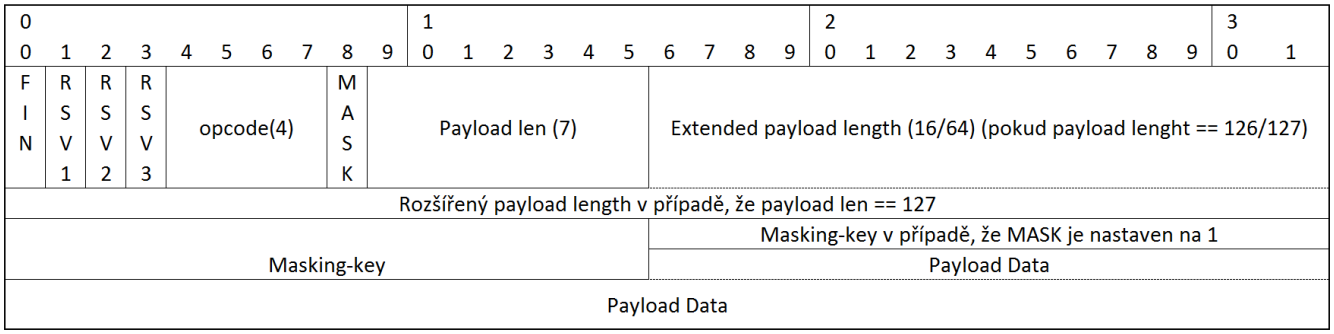
Čas	**Klient - Polling**	**Server - Polling**	**Klient - WebSocket**	**Server - WebSocket**
0 ms	Odeslání 1. požadavku		Povýšení spojení	
50 ms		Přijem 1. požadavku Odeslání 1. dat		Odeslání 1. dat
100 ms	Přijem 1. dat Odeslání 2. požadavku		Přijem 1. dat	Odeslání 2. dat
150 ms		Přijem 2. požadavku Odeslání 2. dat	Přijem 2. dat	Odeslání 3. dat
200 ms	Přijem 2. dat Odeslání 3. požadavku		Přijem 3. dat	Odeslání 4. dat
250 ms		Přijem 3. požadavku Odeslání 3. dat	Přijem 4. dat	Odeslání 5. dat
300 ms	Přijem 3. dat Odeslání n požadavku		Přijem 5. dat	Odeslání n dat

Z tabulky 12 je vidět, že u tradiční metody dotazování je nutné pro každý příjem dat odeslat zvláštní požadavek z klienta. Až po jeho doručení serveru je možné odeslat data zpět. Na druhou stranu u WebSocket je po navázání spojení možná obousměrná komunikace a server nepotřebuje vyžádání na odeslání dat.

3.1.4 Datové rámce

Datové rámce jsou používány k přenosu dat. Do datových rámců jsou zabaleny veškeré zprávy, které chce klient nebo server odeslat. Je povinné před odesláním zprávy data do rámce zabalit. Rámce mohou být odeslány kdykoli po odeslání a přijmutí handshake. Základní podobu rámce ukazuje obrázek 9.

- **FIN** - Pokud je nastaven na 1, pak indikuje, že daný rámec je posledním fragmentem zprávy. První fragment může být zároveň i posledním. Má velikost jednoho bitu.
- **RSV1, RSV2, RSV3** - Rezervované hodnoty pro budoucí verze. V současnosti se používají v případě dohodnutých rozšíření. Musí být nastaveny na 0, pokud není na začátku komunikace dohodnuto rozšíření. Každý má 1 bit.



Obrázek 9: Datový rámec

- **Opcode** - Definuje, o jaký typ posílaných dat se jedná. Typ může být následující:
  - **%x0** - definuje, že se jedná o jeden rámec z vícero.
  - **%x1** - definuje textový rámec.
  - **%x2** - definuje binární rámec.
  - **%x3-7** - tyto jsou rezervovány pro neřídící rámce.
  - **%x8** - značí uzavření spojení.
  - **%x9** - značí ping.
  - **%xA** - značí pong.
  - **%xB-F** - rezervovány pro řídící rámce.

  Pokud je přijat neznámý typ, musí být připojení ukončeno.
- **Mask** - Definuje, zda jsou data maskována. Pokud je hodnota 1, pak maskována jsou. Všechna data posílána od klienta musí být maskována. Velikost je 1 bit.
- **Payload length** - Definuje velikost Payload data. V případě, že je velikost od 0 do 125, pak číslo přímo určuje velikost. Je-li hodnota 126, pak následující dva bajty jsou interpretovány jako 16-bitový unsigned integer, který definuje velikost. Pokud je hodnota 127, pak následujících 8 bajtů je interpretováno jako 64-bitový unsigned integer, kdy nejvýznamnější bit musí být 0. Tento integer pak určuje velikost.
- **Masking key** - Všechny rámce poslané z klienta musí být maskovány 32 bitovou hodnotou odeslanou v rámci. Je-li hodnota nastavena na 1, je klíč obsažen.
- **Payload data** - Payload data obsahuje Extension data a Application data. Jedná se o samotný obsah zprávy.
- **Extension data** - Tato data mohou být nulová, pokud není mezi klientem a serverem domluvené rozšíření. Rozšíření musí definovat velikost, aby bylo možné data oddělit od Application data. Jakým způsobem budou zpracována příjemcem musí být domluveno při předávání handshake.

- **Application data** - Tato data jsou povinná. Velikost se rovná payload length mínus velikost Extension data.

3.1.5 Maskování

Maskování je povinné při přenosu rámců z klienta na server. Každý klientský rámec musí obsahovat 32-bitový maskovací klíč, který obsahuje náhodnou hodnotu, kterou zvolí klient. U každého nového rámce musí být vždy zvolena nová hodnota klíče. Velice důležitá je náhodnost klíče, nesmí být možné jakýmkoli způsobem hodnotu odhadnout.

K maskování i odmaskování je použit stejný algoritmus. Maskuje se po 1 bajtu, tedy 8 bitech. 8 bitů maskovaných dat je 8 bitů originálních dat XOR maskovací klíč modulo 4. Maskovací algoritmus může vypadat následovně:

```
for (int i = 0; i < DECODED.length; i++)
{
   ENCODED[i] = DECODED[i] ^ MASK[i % 4];
}
```

Výpis 3: Maskování dat

3.1.6 Komunikace

3.1.6.1 Odesílání dat Aby bylo možné odeslat zprávu obsahující data, je nutné splnit několik požadavků:

- **Otevřené spojení** - spojení mezi klientem a serverem musí být otevřené. Pokud se v jakékoli části přenosu zprávy stav změní, musí být přenos ukončen.
- **Datové rámce** - odesílaná data musí být zabalena do datového rámce. Více v 3.1.4. Pokud jsou data příliš velká nebo nejsou v danou chvíli všechna k dispozici, je možné je rozdělit do několika samostatných rámců.
- **Opcode** - musí být správně nastaven tzv. opcode. Více v 3.1.4.
- **FIN Bit** - musí být nastaven FIN bit posledního rámce na 1. Více v 3.1.4.
- **Maskování** - pokud jsou data posílána z klienta, rámce musí být maskovány. Více v 3.1.5.
- **Rozšíření** - Pokud bylo mezi serverem a klientem dohodnuto nějaké rozšíření v rámci WebSocket protokolu, je nutné zohlednit i tato rozšíření, dle jeho specifikace.
- **Odeslání** - Je-li rámec hotový a splňuje výše popsaná pravidla, musí být odeslán komunikačním kanálem.

3.1.6.2 Přijímaní dat

- **Otevřené spojení** - příjemce musí naslouchat na daném otevřeném spojení.
- **Datové rámce** - data musí být zpracována jako WebSocket datové rámce, definované dle standardu 3.1.4.
- **Opcode** - z opcode se získá informace o jaký typ dat se jedná. Je-li zpráva fragmentována do více datových rámců, o konci zprávy rozhoduje FIN bit.
- **Rozšíření** - rozšíření, která byla při komunikaci dohodnuta, musí být brána v potaz. Zpracování příchozích zpráv se poté může změnit, kdy před definovanými daty z rámce, se mohou nalézat data z rozšíření.
- **Maskování** - pokud je příjemce server, musí odstranit maskování ze zprávy.

Aby bylo možné přijmout data, musí koncový bod serveru nebo klienta naslouchat na daném připojení. Příchozí data poté musí být zpracována jako WebSocket datové rámce, definované dle standardu. Více v 3.1.4. Poté je nutné z opcode zjistit o jaký typ dat se jedná.

Příklad typické komunikace ukazuje obrázek 10.

3.1.6.3 WebSocket API WebSocket specifikace definuje API, které umožňuje webovým stránkám využít WebSocket protokol. Ten využívá stejné porty jako HTTP a HTTPS, tedy 80 a 443.

Při implementaci jsou důležité čtyři posluchače událostí, které usnadňují obsluhu typických operací při komunikaci.

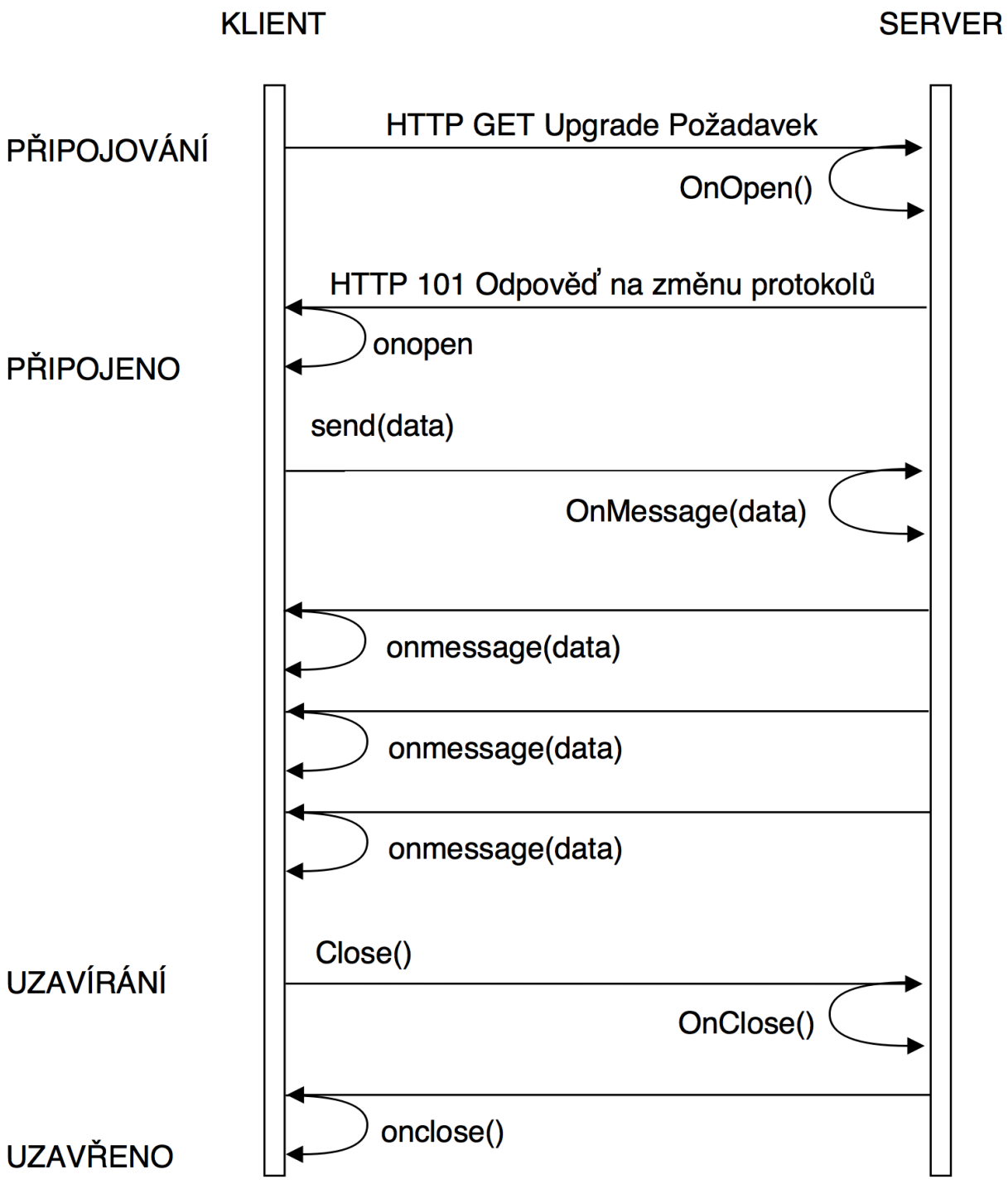
- **OnOpen** - definuje operace, které mohou být provedeny při ustavení spojení.
- **OnMessage** - definuje operace, které mohou být provedeny při doručení zprávy.
- **OnClose** - definuje operace, které mohou být provedeny při ukončení spojení.
- **OnError** - definuje operace, které mohou být provedeny při chybě v přenosu.

Pro posílání dat je pak definována funkce *send()* přijímající jako parametr pole charů.

3.1.7 Bezpečnost a zvládání chyb

WebSocket definuje několik mechanismů pro zvýšení bezpečnosti při komunikaci a ochraně proti útokům.

- **Kontrola původu** - WebSocket kontroluje Origin pole jako ochranu proti jejímu zfalšování. Klient typicky komunikuje se serverem přes webový prohlížeč. Není to ale podmínka a klient může se serverem komunikovat přímo a de facto serveru podstrčit libovolný obsah



Obrázek 10: Komunikace mezi klientem a WebSocket Serverem

Origin. Server by proto neměl automaticky důvěřovat klientovi a věřit, že komunikuje s webovým prohlížečem. V případě, že pole Origin indikuje, že bylo zfalšováno, server musí odeslat HTTP 403 Forbidden status kód.

- **Maskování** - o maskování je více psáno v 3.1.5.
- **Autentizace** - Standard jako takový nedefinuje žádné formy autentizace klienta během předávání handshake. Server ale může použít jakýkoli mechanismus, který je přípustný v HTTP, jako jsou cookies, HTTP autentizace nebo například TLS.
- **Zpracování chyb** - veškerá příchozí data musí být vždy validována jak serverem, tak i klientem. Pokud jsou v kteroukoli chvíli data špatně formátována, porušují specifikaci, tak spojení musí být ukončeno. S ukončením by měl být odeslán i odpovídající kód statusu. Typickým příkladem, kdy může k tomuto dojít, je chvíle, kdy jsou odesílána textová data se špatným formátováním. Specifikace definuje, že textová data musí být v UTF-8 kódování.

3.2 Implementované funkce

K implementaci funkcí serveru je použita knihovna Libwebsocket [4] napsaná v jazyce C, díky čemuž bylo možné začlenění funkcionality vzdáleného ovládání do celého systému, neboť i ten je psán v jazyce v C, respektive C++. Klientská část je poté implementována v jazyce html, javascript, jquery s použitím kaskádových stylů.

3.2.1 Server

3.2.1.1 callback_http

Tato funkce obsluhuje čistý http provoz. Než je připojen první klient, chová se server jako čistý HTTP server. Po povýšení na WebSocket jsou veškerá HTTP volání ignorována.

3.2.1.2 callback_dumb_increment

Tato funkce obsluhuje veškerý WebSocket provoz. Funkce je volána při příchozí zprávě v rámci WebSocket komunikace. Podle toho, o jakou zprávu se jedná, jsou volána následující zpětná volání:

- **LWS_CALLBACK_ESTABLISHED** - definuje, co se stane v případě, že bylo navázáno nové spojení s klientem. V případě řídícího software se jedná o kontrolu, zda je k řídící jednotce připojena kamera a zároveň je získána informace o nastavení panoramata. V případě, že kamera připojena není, je vyslán serverem požadavek na připojení kamery.
- **LWS_CALLBACK_RECEIVE** - zpětné volání je inicializováno v případě, že je klient již připojen a byla z jeho strany odeslána zpráva. Jsou zde parsována získaná data z řídící jednotky o nastavení panoramatu a informace o nastavení kamery. V průběhu focení panoramatu je zpětné volání inicializováno periodicky kvůli aktualizaci dat z focení. Je zde

také obsloužen požadavek na pořízení náhledu, který je následně vykreslen do klientského prohlížeče.

3.2.1.3 protocols

Jedná se o strukturu, která reprezentuje jeden protokol, který bude podporovaný serverem. Protokolů může být definovaných libovolné množství. Pole těchto struktur je poté předáno funkci *lws_create_server*. Struktura může nabývat následujícími parametry:

```
struct lws_protocols {
    const char * name;
    callback_function * callback;
    size_t per_session_data_size;
    size_t rx_buffer_size;
    unsigned int id;
    void * user;
};
```

Výpis 4: Struktura protokolu

- **name** - jméno protokolu. Je nutné, aby odpovídalo funkci, která je definována klientem v Javascriptové funkcí *new WebSocket(url, 'protocol')*.
- **callback** - definuje funkci, která bude zpracovávat zpětná volání, která zajistí obsluhu klientských požadavků.
- **per_session_data_size** - určuje, jak velké množství paměti bude každému novému klientovi alokováno pro danou relaci. Po ukončení spojení je paměť automaticky uvolněna.
- **rx_buffer_size** - definuje maximální velikost rámce.
- **id** - tato položka může obsahovat informaci o daném protokolu. Není povinná.
- **user** - uživatelem poskytnutá data. Knihovna jako taková je ignoruje.

Nejsou povinné všechny parametry, ale pouze první tři z nich. Reálná implementace vypadá následovně:

```
static struct lws_protocols protocols[] = {
  {
    "http-only", // name, first protocol must always be HTTP handler
    (lws_callback_function*)callback_http, // callback
    0            // no per_session_data_size
  },
  {
```

```
        "dumb-increment-protocol", // protocol name, this is WebSocket handler
        (lws_callback_function*)callback_dumb_increment, // callback
        0                          // no any per session data
    },
    {
        NULL, NULL, 0 //end of list
    }
};
```

Výpis 5: Implementace protokolu

3.2.1.4 StartRemoteServer

Funkce spouští server vzdáleného ovládání. Má proto vyhrazené vlastní vlákno, inicializované z hlavního programu řídícího software.

V těle funkce je definováno několik důležitých parametrů:

- **port** - Port, na kterém bude server naslouchat. Defalutně se používá port 5000.
- **context_info.protocols** - Odkazuje na strukturu protokolů. Více v 3.2.1.3
- **context_info.extensions** - Určuje, zda bude dohodnuto nějaké rozšíření nad rámec protokolu WebSocket. Pokud ne, je nastaveno na NULL.
- **context_info.ssl_cert_filepath** - Je-li využito zabezpečení pomocí ssl, definuje se cesta k certifikátu.
- **context_info.ssl_private_key_filepath** - Definuje cestu k privátnímu klíči ssl zabezpečení. Není-li využito, nastaví se na NULL.
- **lws_create_context** - Tato funkce vytváří serverový socket pro naslouchání a stará se o inicializaci dle nastavených parametrů, které jsou popsány výše. Po inicializaci funkce vrací strukturu *lws_context **, která reprezentuje server. Zpočátku je server čistě HTTP, než dojde k povýšení při výměně handshake s prvním klientem.
- **lws_service** - Funkce, která obslouží veškeré požadavky klientů i serveru. Funkce přijímá všechny nové klienty na již vytvořený server a volá odpovídající zpětná volání na požadavky klientů. Přijímá dva parametry. Prvním je kontext, který byl vytvořen funkcí *lws_create_context*. Druhým parametrem je poté časový limit, po kterém bude funkce znovu spuštěna. Protože server běží v samostatném vlákně, tak není problém, že je vlákno při čekání na události blokováno. Vhodným nastavením časového limitu je možné dosáhnout rychlé reakce na události a přitom nevytěžovat procesor řídící jednotky, když je server v režimu čekání na požadavky.

3.2.1.5 WriteString

Funkce která umožňuje odeslat řetězec stringů klientovi. Před samotným odesláním je nutné provést přípravné operace. Je žádoucí si dopředu alokovat potřebnou paměť.

Ta musí mít následující velikosti: LWS_SEND_BUFFER_PRE_PADDING + velikost_zprávy + LWS_SEND_BUFFER_POST_PADDING.
Každá odeslaná zpráva musí být odeslána jako WebSocket rámec, jak je popsáno v 3.1.4. LWS_SEND_BUFFER_PRE_PADDING a LWS_SEND_BUFFER_POST_PADDING tak rezervují místo v paměti pro všechny náležitosti, které jsou při odeslání nutné nastavit.

K samotnému odeslání pak slouží funkce *lws_write*, která kromě dat k odeslání a velikosti dat předává i odkaz na strukturu popisující aktivní spojení.

3.2.2 Klient

Klientská aplikace slouží k ovládání řídicí jednotky motorizované panoramatické hlavy a nastavení připojeného fotoaparátu. Díky použití technologie WebSocket a zvolené implementaci v jazyce html, javascript a jquery se předpokládá vysoká variabilita možných zařízení, z kterých půjde aplikaci použít. Jakékoli zařízení z integrovanou technologií Wi-Fi a novějším internetovým prohlížečem, by mělo být schopno aplikaci korektně provozovat. Podporu protokolu WebSocket u nejpoužívanějších internetových prohlížečů ukazuje tabulka 13.

Tabulka 13: Podpora WebSocket v prohlížečích, platná k 27.3.2017

IE	Edge	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android prohlížeč	Chrome na Androidu
			49						
			55					4.4	
		51	56			9.3		4.4.4	
11	14	52	57	10	43	10.2	nic	53	56

3.2.2.1 websocketInit

Tato funkce v klientské implementaci naváže spojení se serverem. Proběhne výměna handshake a pokud vše proběhne v pořádku, tak je jako návratová hodnota vrácena struktura *websocket*, která popisuje navázané spojení. Implementace je velice přímočará a neobsahuje mnoho řádků kódu:

```
function websocketInit() {
        window.WebSocket = window.WebSocket || window.MozWebSocket;
        websocket = new WebSocket('ws://192.168.100.7:5000', 'dumb-increment
            -protocol'); // adresa serveru:port a nazev protokolu
        websocket.onopen = function () {
            $('h1').css('color', 'green');
        };
```

```
        websocket.onerror = function () {
            $('h1').css('color', 'red');
        };
        return websocket;
    }
```

Výpis 6: Klientská inicializace WebSocket

3.2.2.2 getParam

Při navázání spojení mezi serverem a klientem jsou ze serveru automaticky odeslány parametry nastavení panoramatického snímku a je-li připojena kamera, tak i nastavení její. U připojené kamery nejsou odeslány pouze aktuální hodnoty nastavení, ale i všechny možné hodnoty, které nastavit lze. Uživatel pak při nastavení jednoduše vybere z již předpřipravených hodnot. Data jsou formátována jako pole charů a je proto nutné na straně klientského programu data parsovat. K tomu právě slouží funkce *getParam*.

Přijímá tři parametry. Prvním je pole, do kterého budou vyparsované hodnoty uloženy, druhým je poté již částečně parsovaný buffer příchozích zpráv a třetím je informace o tom, co se bude parsovat.

Příklad použití při získání aktuální hodnoty citlivosti a všech možných hodnot nastavení: *getParam(iso, res[4], "iso");*.

3.2.2.3 sendCamParam

Uživatel má z klientského prohlížeče možnost nastavovat parametry panoramatu a připojeného fotoaparátu. Poté, co je vše nastaveno, je třeba hodnoty přenést z klienta na server, kde jsou dále zpracovány řídícím programem motorizované hlavy. Funkce *sendCamParam* vyparsuje uživatelem nastavené hodnoty fotoaparátu a odešle je na server.

3.2.2.4 sendPanoParam

Funkce má podobné vlastnosti jako *sendCamParam* s tím rozdílem, že parsuje hodnoty nastavení panoramatu.

3.3 Uživatelské rozhraní

Uživatelské rozhraní vzdáleného ovládání umožňuje nastavení všech parametrů, dostupných při ovládání přes tlačítka a displej na hlavním ovládacím panelu motorizované hlavy. Při návrhu byla zohledněna variabilita možných zařízení, na kterém klientská aplikace bude spuštěna. Vzhled je proto jednoduchý s použitím generických prvků tak, aby byla zajištěna co největší kompatibilita.

3.3.1 Úvodní obrazovka

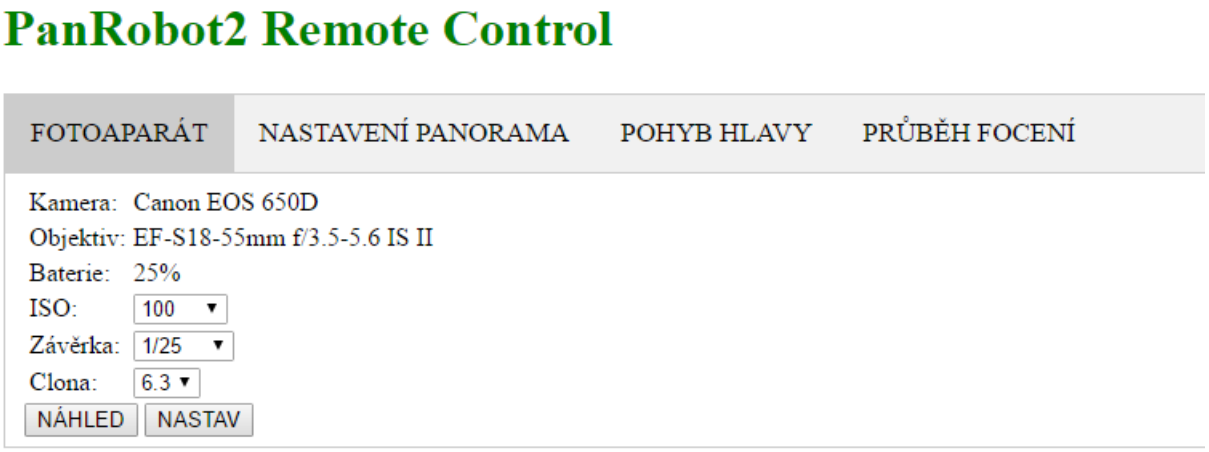
V případě otevření klientské aplikace se mohou zobrazit dvě odlišné úvodní obrazovky. V případě, že je korektně ustaveno spojení, je barva názvu aplikace zelená, v opačném případě, kdy spojení nebylo navázáno, je červená, jak ukazuje obrázek 11 .

Obrázek 11: Navázané/Nenavázané spojení mezi klientem a serverem

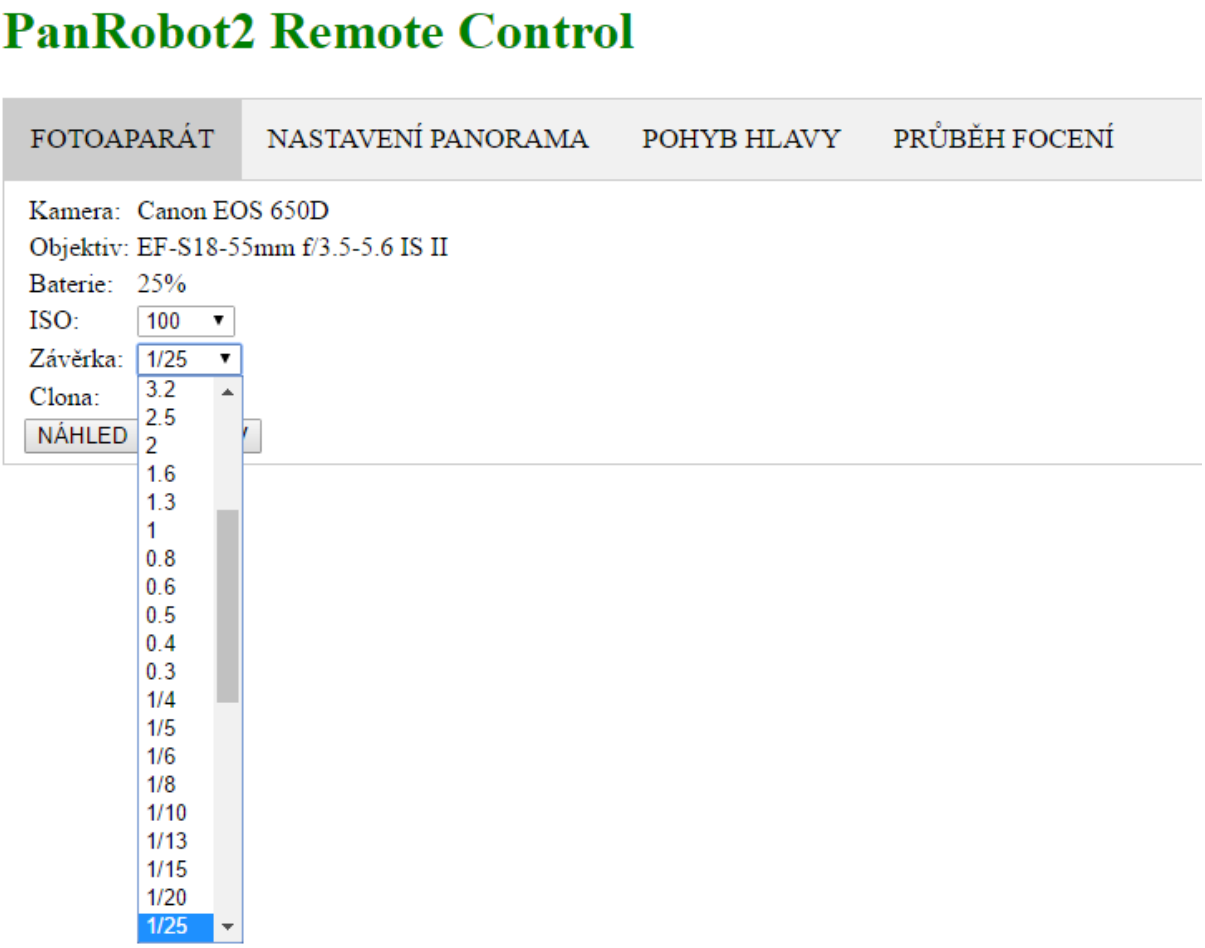
3.3.2 Fotoaparát

V případě, že je korektně připojen fotoaparát a řídící software s ním navázal spojení, je možné na kartě *FOTOAPARÁT* získat základní informace o fotoaparátu včetně stavu baterie a dále nastavit tři základní parametry, které jsou relevantní při focení panoramatu: ISO, Závěrku a Clonu.

Parametry je možné nastavit z výsuvného menu, jak ukazuje obrázek 12 a 13, které obsahuje jen a pouze ty hodnoty, které fotoaparát skutečně umožňuje nastavit. Více v kapitole 2.2.8.



Obrázek 12: Přehled základních údajů o fotoaparátu



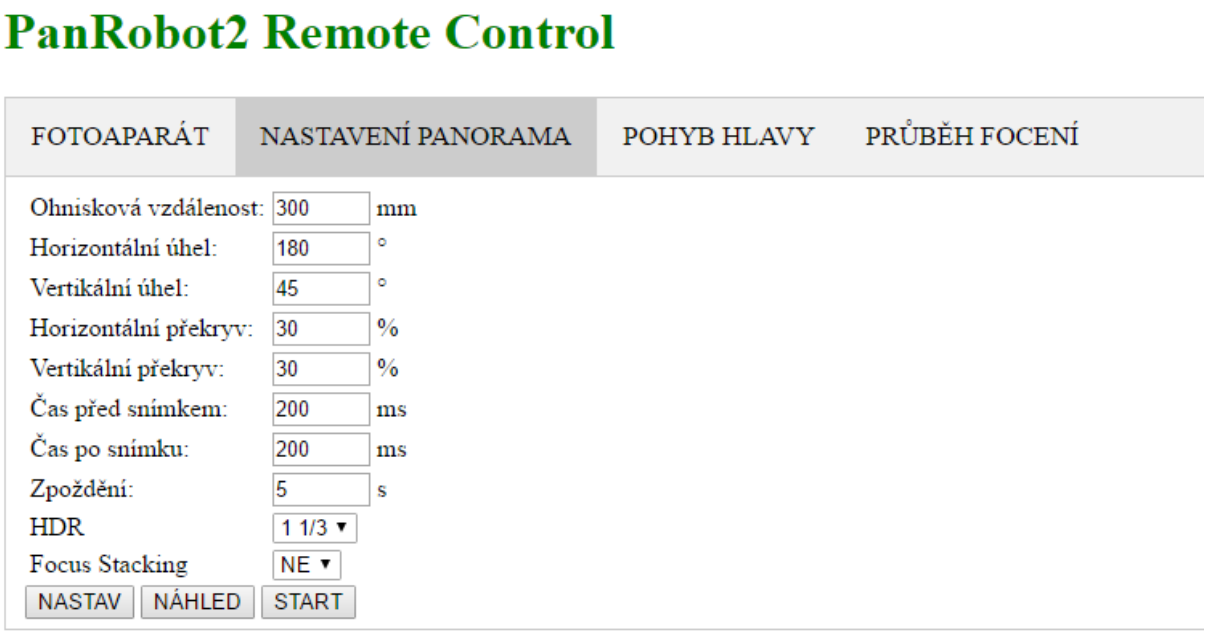
Obrázek 13: Výběr parametru fotoaparátu

Pomocí tlačítka NÁHLED je možné pořídit rychlý náhled snímané scény a přímo jej zobrazit v klientské aplikaci. To je praktické při nastavování úhlů záběru panoramata, jakož i pro kontrolu správně nastavené expozice, protože náhledový snímek je pořízen s aktuálně nastavenou expozicí. Opakovaným stiskem tlačítka jsou pořizovány nové náhledy. Viz. obrázek 14.

Obrázek 14: Karta FOTOAPARÁT s pořízeným náhledem

3.3.3 Nastavení Panorama

Záložka *NASTAVENÍ PANORAMA* umožní nastavit parametry potřebné k vyfocení požadovaného panoramatického snímku. Je možné také nastavit pokročilé funkce HDR a Focus Stacking. Po dokončení nastavení je potřeba stisknout tlačítko NASTAV, aby byly hodnoty přeneseny do řídící jednotky. Tlačítko START poté spustí celý proces focení panoramatu. Viz. obrázek 15.



Obrázek 15: Přehled obrazovky nastavení fotoaparátu

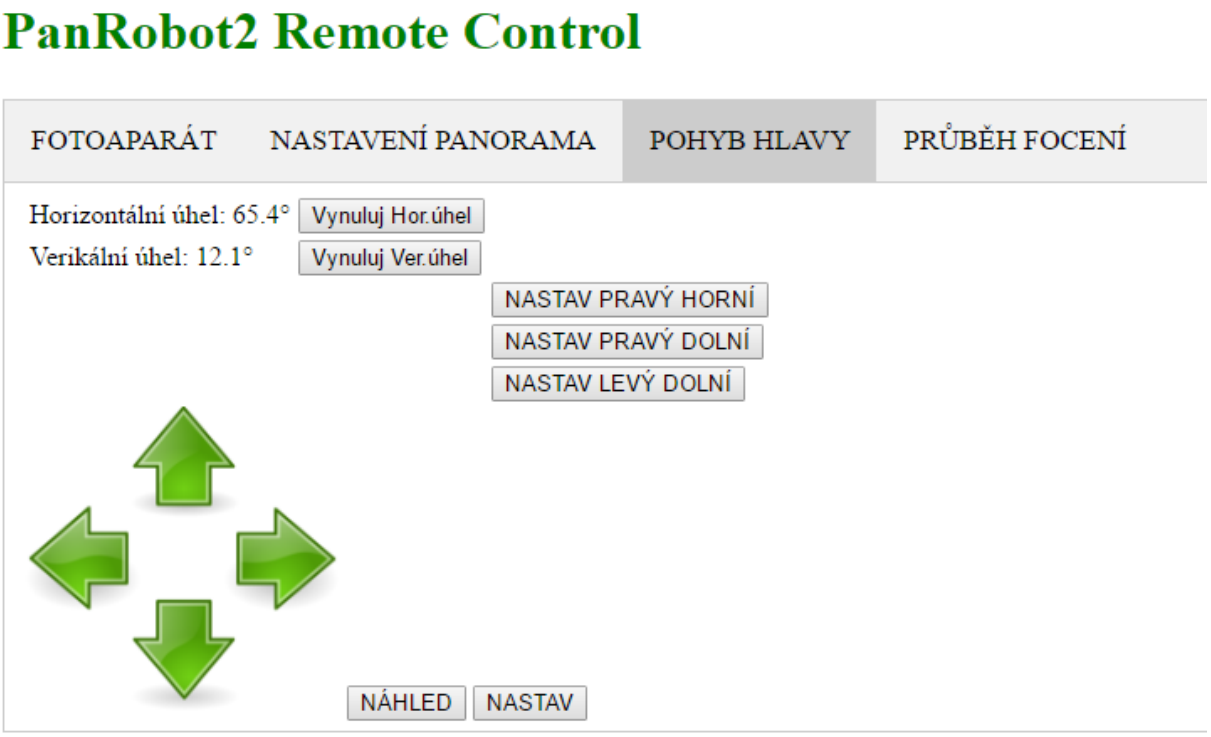
3.3.4 Pohyb Hlavy

Zde je možné dálkově ovládat pohyb dvou krokový motorů a nastavit tak horizontální a vertikální úhel, které určuji rozsah záběru, který bude celkový snímek mít. Jak to vypadá v klientské aplikaci zobrazuje obrázek 16. Chce-li uživatel nastavit jednotlivé úhly, pak může postupovat takto:

1. Nastavit panoramatickou hlavu pomocí stisknutí příslušných šipek do pozice, která odpovídá pravému hornímu rohu budoucího panoramatického snímku. Potvrzení pozice provede stisknutím tlačítka *NASTAV PRAVÝ HORNÍ*

2. Pohybem hlavy změní pozici na pravý dolní roh a volbu potvrdí stiskem tlačítka *NASTAV PRAVÝ DOLNÍ*. V tomto okamžiku je již znám vertikální úhel a uživatel jeho hodnotu může zkontrolovat v informačním poli. Stejně se změní i hodnota na kartě *NASTAVENÍ PANORAMATA*.

3. Posledním krokem je poté nastavení hlavy do pozice levého dolního rohu a potvrzení příslušným tlačítkem. V tomto okamžiku je znám i celkový horizontální úhel.



Obrázek 16: Obrazovka pohybu hlavy

3.3.5 Průběh focení

Po stisku tlačítka START na kartě *NASTAVENÍ PANORAMA* jsou na této kartě zobrazeny informace o průběhu focení. O jaké informace se jedná ukazuje obrázek 17.

Obrázek 17: Obrazovka průběhu focení

4 Testování

U systému jako je tento, je jednou z klíčových vlastností spolehlivý chod. Jakýkoli výpadek během focení panoramatického snímku má za následek znehodnocení požadovaného výsledku jako celku, tak i nemalou ztrátu času. Výrazným vylepšením z hlediska spolehlivosti je bezpochyby aktivní spojení mezi řídící jednotkou motorizované hlavy a fotoaparátem. V předchozí verzi bylo spojení pouze pasivní a řídící program neměl možnost ověřit, že snímek, který požadoval vyfotit, skutečně vyfocen byl. Hrozilo tak riziko, že u panoramat, kdy je foceno několik stovek nebo i tisíců snímků bude jeden nebo více chybět, což má za následek nepoužitelnost všech ostatních snímků. U aktivního spojení existuje zpětná vazba, která zajistí téměř 100% jistotu, že snímky opravdu vyfoceny byly.

Pro nový systém byl navrhnut i zcela nový plošný spoj s využitím všech získaných zkušeností s používáním předchozí verze systému. I to bude mít jistě na spolehlivost pozitivní vliv.

4.1 Reálný provoz

Dne 27.3.2017 byl proveden reálný test nového systému. Z hory Huk, na kterém se nachází dnes již nevyužívaný vykrývač televizního signálu, byla za použití horolezeckého vybavení umístěna panoramatická hlava, která pořídila snímek města Vrbna pod Pradědem. Umístění i focené město bylo vybráno záměrně z toho důvodu, že ze stejného místa byl testován i předchozí systém. Bylo tak možné porovnat oba systémy a nalézt případné nedostatky.

V plné míře se ukázala výhoda možnosti vzdáleného ovládání, neboť panoramatická hlava byla umístěna na těžko přístupném místě. Po připevnění na místo se celé nastavení a spuštění panoramatické hlavy provedlo z paty vysílače, ve vzdálenosti 8 metrů od zařízení. Jako klientské zařízení byl použit běžný mobilní telefon se systémem Android 5.1 s nainstalovaným internetovým prohlížečem Chrome.

V době psaní této práce ještě nebyl kompletní snímek hotov. Parametry při focení a předpokládané parametry po dokončení snímku jsou následující:

- **Ohnisko objektivu** - 600 mm při kinofilmovém políčku, 960 mm po přepočtu na velikost snímače použitého fotoaparátu. Oproti předchozímu snímku se jedná o dvojnásobné ohnisko. U minulé verze systému nebylo objektivně možné využít při vytváření takhle velkého panoramatu ohnisko větší, než 300 mm. Důvodem je samotná délka focení, která způsobí takové změny světelnosti a vrhání stínů, že zpracovaný snímek nevypadá konzistentně. Díky aktivnímu propojení fotoaparátu a preciznější spolupráci s motory panoramatické hlavy bylo možné celé fotografování výrazně zrychlit.
- **Horizontální úhel záběru** - 195 °
- **Vertikální úhel záběru** - 34 °
- **Počet vyfocených snímků** - 2618

- **Doba focení** - 65 minut. Předchozí panorama bylo foceno 80 minut s počtem snímků 1200.
- **Velikost na disku (Součet jednotlivých RAW souborů)** - 65.5 GB
- **Počet dílčích snímků pro webové prohlížení** - Dle výpočtu by počet snímků měl být minimálně 10 000 000.

Po dokončení zpracování bude snímek publikován na adrese `<http://www.gpix.eu>`

4.2 Spotřeba energie

Nejobávanějším negativem, které mohlo přinést přechod z mikrokontroleru AT-Mega na RaspberryPi bylo snížení výdrže na baterii. Před samotným zahájením vývoje a definitivním rozhodnutím, že bude použito Raspberry Pi 3, byly provedeny série testů a výpočtů.

Od samotného začátku bylo jasné, že pro provoz řídícího software nebude potřeba využití plného výkonu čtyř jader. Do hry tedy vstupovala možnost snížení maximální procesorové frekvence za účelem snížení celkové spotřeby. Tomuto kroku nahrávala i skutečnost, že doba vykonávání procesu není odvislá od výkonu procesorů. Základní podmínkou bylo, aby celý systém dokázal na jedno nabití akumulátoru provést kontinuální focení panoramatu po dobu dvou hodin, a to včetně zapnutého vzdáleného ovládání.

Při použití akumulátoru s kapacitou o hodnotě 5Ah, nesmí průměrná spotřeba proudu pro dvouhodinový provoz překročit 2.5Ah. Jedná se samozřejmě o teoretické hodnoty, kapacita baterie je jistě menší, než udává výrobce. Navíc akumulátor byl již používán se starší verzí panoramatické hlavy a bylo provedeno několik desítek nabíjecích cyklů, to kapacitu baterie také snížilo. Aby byla splněna podmínka dvouhodinového provozu a zároveň zůstala rezerva, upravil jsem parametry tak, že akumulátor má kapacitu 4Ah a průměrná spotřeba proudu nesmí přesáhnout 1.8Ah.

Omezením počtu využitelných jader a omezením maximální frekvence procesoru bylo dosaženo snížení průměrné spotřeby z 350mAh na 290mAh. Největším odběratelem proudu tak zůstaly dva krokové motory, ale omezením maximálního možného odběru a optimalizací software se spotřeba proudu u motoru pro horizontální osu snížila z 400mAh na 80mAh a u vertikálního motoru z 580mAh na 250mAh. Souhrnný průměrný odběr proudu celého systému po optimalizaci tak činí 680mAh.

Spotřeba oproti plánu je tak více než 2.5 krát menší, což umožní realizovat i ty největší panoramatické projekty s dobou focení v řádu hodin.

5 Závěr

Hlavním cílem této práce bylo rozšířit stávající systém pro pořizování panoramatických snímků. Veškeré potřeby rozšíření vycházely z praktického používání předchozí verze řídícího systému, tak i z testování komerčních produktů stejného zaměření.

Podařily se splnit všechny cíle zadání. Při implementaci vzdáleného ovládání byla oproti původnímu návrhu provedena po dohodě s vedoucím práce změna, kdy se upustilo od vývoje aplikace pro operační systém Android s tím, že bude využita technologie WebSocket, která umožnila výrazným způsobem rozšířit počet zařízení, na kterém je možné klientskou aplikaci provozovat. Pokročilé funkce jako HDR a Focus Stacking posunuly možnosti řídícího systému a motorizované hlavy na vysoce profesionální úroveň. Dle dostupných informací se jedná o jediný systém, který obě tyto funkcionality nativně podporuje. U funkcionality Focus Stacking je nutno zmínit, že je ve chvíli psaní této práce dostupná jen pro digitální fotoaparáty značky Nikon. V budoucím vývoji se ovšem počítá s přidáním podpory i pro vybrané fotoaparáty Canon.

Stále existuje velký prostor pro rozšíření funkcionalit. Plánovaná je podpora pokročilého fotografování objektů hlubokého vesmíru s přesným automatickým pozicováním dle volně dostupných internetových databází vesmírných objektů a režimem pro sledování vybraného objektu při jeho pohybu po obloze.

Literatura

[1] *LOMBARDI*, Andrew, 2015. *Websocket: Lightweight Client-Server Communications*, Vyd. 1. O'Reilly Media. [cit. 2017-03-10]
ISBN 9781449369279

[2] *RUSSEL*, Jesse, *COHN*, Ronald 2016. *Picture Transfer Protocol*, Vyd. 1. Book on Demand, Miami. [cit. 2017-03-10]
ISBN 9785512153949

[3] libgphoto2
URL: <`https://github.com/gphoto/libgphoto2`> [cit. 2017-03-10]

[4] libwebsocket
URL: <`https://libwebsockets.org/`> [cit. 2017-03-23]

[5] Obadal, Tomáš; *Řídící systém motorizované panoramatické hlavy*
URL: <`https://dspace.vsb.cz/handle/10084/104156`> [cit. 2017-04-01]

[6] International Organization for Standardization; *ISO 15740:2013*
URL: <`https://www.iso.org/standard/63602.html`> [cit. 2017-04-01]

[7] caniuse.com; *WebSocket browsers support*
URL: <`http://caniuse.com/#feat=websockets`> [cit. 2017-03-27]

A Obsah přiloženého datového nosiče

- text diplomové práce
- zdrojový kód řídícího software motorizované panoramatické hlavy
- zdrojový kód klientské aplikace pro vzdálené ovládání
- zdrojový kód mikrokontroleru pro řízení krokových motorů